Towards a Toolkit for Free Living Wearable Development

Blaine Rothrock
Alexander Curtiss
Juyang Bai
blaine@northwestern.edu
email@northwestern.edu
juyangbai2023@u.northwestern.edu
Northwestern University
Evanston, IL, USA

Josiah Hester
Georgia Institute of Technology
Atlanta, GA, USA
josiah@northwestern.edu

ABSTRACT

Real-world Data collection and analysis is a significant pain point in wearable device development, requiring multidisciplinary skills in: embedded systems, application development, data science, and domain expert knowledge. In this work, we first build motivation based on previous experiences in wearable development, then introduce a toolkit for data collection and iterative development to reduce engineering efforts for free living experimentation of wearable devices. This toolkit utilizes Bluetooth Low Energy and an adaptive mobile application to help researchers quickly test new hardware, collect meaningful data, and assist in developing embedded algorithms with minimal intermediary code changes. We demonstrate the utility of our toolkit by collecting data in-the-wild from multiple sensors using a prototype wearable and a ground truth heart rate sensor. In addition, we demonstrate the toolkit's capabilities with a baseline throughput test. Finally, we show how this tool has helped in early development of a new custom device. Our work is released as open source and welcomes contributions in an effort to broaden the tookit's utility for the wearable research community.

KEYWORDS

Wearables; Health; Machine Learning

ACM Reference Format:
Blaine Rothrock, Alexander Curtiss, Juyang Bai, and Josiah Hester. 2022. Towards a Toolkit for Free Living Wearable Development. In *Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp/ISWC '22 Adjunct), September 11–15, 2022, Cambridge, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3544793.3560399

1 INTRODUCTION

Wearable devices that track our everyday lives are becoming commonplace. However, collecting sensor data from wearables is challenging and creates significant engineering overhead [2], even when merely reproducing well-established methods. In exploring new human sensing modalities, using consumer wearable devices is limiting due to proprietary hardware or software making it difficult to customize or integrate into custom pipelines. Handling relatively high-frequency sensor data for developing new algorithms is particularly challenging. Doing so requires developing new hardware and the scaffolding for testing that hardware from scratch. This sort of effort blurs the lines between hardware engineering, embedded software development, application development, and data science. Developing sensors and data pipelines for wearable research is daunting and can lead to long development runways in establishing new contributions to wearable sensing.

Further complicating the design space, wearables are intended for free-living environments, requiring robustness and efficiency to perform long-term and with minimal burden. Devices must be reasonably unobtrusive while accounting for a user's activity, which introduces noise to sensing data—without a flexible development workflow, experimenting in the real-world leads to slow iterations for development teams. For example, assume a team is developing a new algorithm for heart-rate detection but is unsure of a specific threshold parameter for onboard filtering. Detection may work well while stationary, but moving about the world introduces scenarios when detection requires additional processing or is not impossible. When computed results do not align with ground truth comparison, raw-sensor data may need to be collected and analyzed before identifying the issue. Due to size constraints and placement, wearables often have little or no user interface; therefore, alternating between onboard functionality may require re-flashing the device, and raw-sensor data collection may not be possible outside the lab. Generally, this workflow is slow and limits iterative development by tethering wearable development to overly controlled environments, which is not ideal given their intended application.

In this work, we present the initial development of a toolkit which encompasses a communication specification for BLE and a mobile application to help in this process. We have found that custom hardware can be worn and tested as part of initial development quickly after board bring-up. First, we describe the pain points specifically in a user scenario. Second, we outline requirements compiled from previous projects and collaborations, building motivation for the tool. Then walk through our implementation, followed by a demonstration of collecting photoplethysmography (PPG) and accelerometer data. Finally, we present a throughput test of our tookit, and exhibit how findings of our tests assist in

UbiComp/ISWC '22 Adjunct, September 11–15, 2022, Cambridge, United Kingdom

ACM ISBN 978-1-4503-9423-9/22/09…$15.00
https://doi.org/10.1145/3544793.3560399

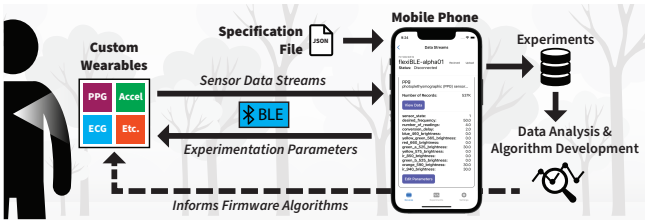


Figure 1: A overview of development workflow using the toolkit, a prototype wearable device communicates with the mobile phone application, sending sensor data and managing parameters defined in a specification file. The mobile app records the data in experiments and organizes it for data analysis. Analyzing the sensor data near-device informs algorithm development for biometrics, which can later be deployed on-device. This process repeats for iterative development, collecting, and experimenting in free-living environments.

iterations of our firmware. Our tool kit is released open source on Github[1], and is open to contributions.

2 MOTIVATION AND BACKGROUND

In developing FaceBit [3], the authors of this work faced challenges in developing algorithms for novel wearable sensing. FaceBit is a custom-designed wearable device deployed inside a face mask, motivated by the demand for personal protective equipment for clinicians early in the COVID-19 pandemic. The device developed methods for measuring mask state (on/off), heart rate, respiration rate, and mask fit. However, given the placement of the device inside a face mask, these metrics required tedious iterations before they were ready for in-the-wild testing. Furthermore, in addition to reporting metrics, FaceBit is designed for low-power consumption to enable multiple uses without recharging or a battery change, further complicating development cycles.

Throughout the development process, the authors maintained two firmware versions for the FaceBit device: a version for collecting raw sensor data (development) and a version that computed metrics onboard (application). The development firmware's goal was to collect sensor data to conceptualize new algorithms, which informed the development of embedded algorithms in the application firmware. However, the development firmware consumed far more energy with continuous BLE streams to a Desktop application. In contrast, the application firmware maintained low consumption sending small data packets roughly every two minutes. In addition, the companion desktop and iOS application had code to handle the different configurations separately and often required updates alongside firmware changes. The team developed embedded algorithms by analyzing raw data from in-lab using Python to identify desired signals, then implementing the prospective algorithm in the firmware. Development iterations require first collecting metrics against a ground truth device, comparing, and often switching between development and application firmware. This process limited the flexibility in development and created long road to conducting evaluations in free-living scenarios.

This experience is the primary motivation for the toolkit presented in this work. As the authors plan new wearable contributions, the need to ease the developmental burden became clear. Shrinking the development runway to experimenting with new devices in the wild and allowing more flexibility for experimentation is a significant hurdle to contributions in the wearable health community. Figure 2 outlines the ideal development workflow with the assistance of our toolkit.

3 RELATED WORK

Platforms exist that focus on data collection of human activity data through wearable and serve as motivation for this work. mCerebrum [7] is a platform for efficient high-frequency data collection and analysis to identify human biomarkers. AWARE [5], RADAR-Base [11], and Raproto [12] focus on data collection from consumer devices. The Digital Biomarker Discovery Pipeline [1] similarly focuses on analysis pipelines of various consumer devices. Finally, SenseCollect [2] motivates the design of platforms for easier Human Activity Recognition by outlining challenges in research teams collecting data and designing a system for data collection from Android watches over WiFi. The primary focus of these works is consumer wearables; we aim to develop custom hardware and form factors that allow for novel sensing mechanisms not currently available.

Many Software-as-a-Service (SaaS) platforms exist to collect sensor data from Internet-of-Things (IoT) devices. These include Particle, Google Cloud IoT, IBM IoT Solutions, and others. These platforms usually require subscriptions and are geared for large enterprise deployments, but more importantly, they require internet connectivity and that data passes through a centralized service. Open source solutions like ThingsBoard and AdaFruit IO are other options better suited in the "maker" space but still require internet connectivity. Wearables primarily utilize Bluetooth Low-Energy (BLE) for low energy consumption and localization. Therefore, these centralized solutions still need a gateway, like a mobile phone, to send data to remote services. Utilizing such tools can be helpful, especially in wide-scale monitoring, but we find using them inefficient for quick iterations since BLE protocols are still required.

4 REQUIREMENTS

Drawing from small-team, wearable health research and development, informed by the development of FaceBit [3], NeckSense [13], and Amulet [6] our goal is to reduce the data burden of developing prototypes and decrease the time tethered to in-lab testing and development. We claim that the faster testing can occur in the wild, the more robust wearable systems will be for their intended purposes. For this context, we define *in-the-wild*, *free-living*, and *real-world* to refer to the collection of data to aid in prototype development while performing any activity considered nominal to an intended wearer. Here we outline the requirements to achieve these goals from experience, which has motivated our initial development.

4.1 Data Collection

The most crucial component of wearable prototyping is data integrity. This tool must reliably collect data from multiple devices

[1]https://github.com/Flexi-BLE

with multiple sensors at frequencies up to 500Hz to support raw sensor data collection with low resource consumption for analysis and algorithm development. Managing free-living experiments in real time requires wireless data collection over Bluetooth, WiFi, or cellular, often collecting data on mobile devices or gateways. Local data storage is viable, with an onboard SD card or non-volatile flash memory. Although, local storage can waste time, space, resources, and participation since functionality and data reliability is unknown until post-analysis. Additional to raw sensor data, this tool must allow for the integration of multiple devices and provide a generalized interface for time-series data collection, allowing for data of varying types and sizes.

4.2 Flexibility

Embedded devices have limited programming interfaces, especially when designing for wearable form factors that often lead to extremely limited or non-existent user interfaces. Updating firmware on a wearable, for example, to change a parameter or swap functionality, requires a connection to a computer, therefore tethering early testing to in-lab. To limit this, we need applications that adaptively respond to runtime updates. Updating embedded parameters should be achievable without code changes or, at most, only require a single code-base to be updated when in active development. We propose that the firmware primarily drives the functionality, and anything upstream adapts to its changes. The firmware must expose configurable parameters and provide a medium for structurally defining these parameters. Examples of such parameters include; toggling functionality, changing sensor frequency or sensor-specific configurations, or even swapping or reconfiguring embedded tasks and workflows.

4.3 Data Organization, Ownership, and Analytics

Structured data collected from various devices in a way that is easily exported and ingested for data analysis is a necessity. All time-series data should be comparable and indexed based on accurate timestamps relative to the sensor. This data must be collected locally (on-person), requiring no internet connectivity to ensure ownership and privacy, but also be organized in a way for easy exportation to centralized services. Finally, during data collection, research requires the functionality to record high-level data about their experimentations, such as notes about particular points in time along with detailed logs of sensor state for reproducibility.

5 IMPLEMENTATION

Motivated by our requirements, we outline a guideline for implementing BLE on embedded firmware, a JSON device specification schema, and a mobile application. The features the core functionality for the toolkit. Except for sensor-specific drivers and identifiable data, the code for this implementation is available on GitHub.

5.1 Hardware and Firmware Development

We envision this toolkit to be agnostic to hardware and firmware, with the exception of requiring BLE for communication where the wearable devices serve as a peripheral role and the mobile application the central. System on chip (SoC) microcontrollers have

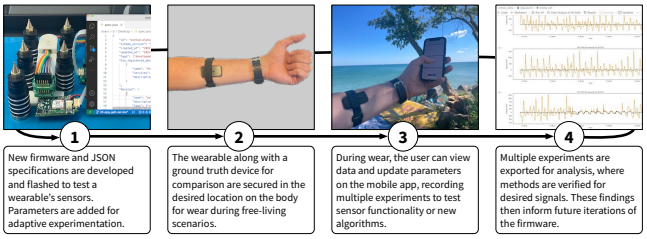


Figure 2: Overview of the ideal workflow for developing novel wearables in free-living scenarios. Developers deploy firmware with adaptable parameters, then adjust them in-the-wild to perform diverse experiments without the need for a computer. Data is later analyzed alongside experimentation metadata to inform future firmware iterations.

small footprints with built-in BLE and well-supported APIs. For our demonstration, we use a Nordic nRF5340 SoC running the Zephyr real-time operating system (RTOS). Zephyr provides an extensive embedded C/C++ API for using BLE, and we provide example code on Github for the off-the-shelf nRF5340 Dev Kit. Bluetooth SIG strictly manages BLE protocols; therefore, our implementation is possible on any BLE-supported system. We assume BLE version 4.2 or greater, which allows for PHY 2M and data length extension enabling higher bandwidth than previous versions. Our implementation is outlined in figure 1.

Supported firmware requires a series of custom Generic Attribute (GATT) services and characteristics to enable developers to easily add new sensors and configurations with arbitrary data. First programmers are required to define a primary service, referred to as the *Info Service*, providing universal characteristics. The Info Service is included in the BLE advertisement packet, identifying the device to the mobile application. Currently, the Info Service holds a single characteristic for a reference epoch time. At the time of connection, the central (mobile phone) writes the current Unix Epoch time, and the device stores a system-time reference to central time, ideally keeping track in an onboard real-time clock (RTC). However, system uptime based on the CPU clock can also suffice. The reference time allows for time synchronization in streaming sensor data and can be periodically updated by the central as needed. Epoch Time is referenced in milliseconds as a 64-bit unsigned integer, where the reference time is 32-bits, technically requiring an update approximately every 50-days. Data sent to the mobile phone converts reference times back to UTC. We saw this system as a viable option in place of the BLE registered Current Time Service, which uses calendar time.

To manage data collection and parameters configuration, additional services defined for each sensor, called *Data Streams*, are added by firmware developers. Data Streams include two characteristics; configuration (read/write) and data (notify). The configuration characteristic is a developer-defined byte array that stores any parameters related to the sensor. The firmware reads from these configurations at run time to perform or guide operations. The data characteristic notifies the central of new sensor data. The first 4 bytes of the data array (unsigned 32-bit integer) contains a *anchor*

time which is the time in milliseconds since the Info Service reference time. The remainder of the array includes developer-defined data packets for batching sensor data, with an appended value after each point indicating the time elapsed since the last value. For example a accelerometer's sensor's data packet could include 3-values for x, y, and z axes, the bytes size depends on the required precision. In BLE version 4.2 or greater, the data-length extension allows for notifying characteristics up to 256 bytes, allowing for greater throughput due to requiring fewer packets and, therefore, less overhead.

5.2 Device Specification

The organization of the firmware, including GATT service details and specifics on data stream organization, is structured into a JSON file. This JSON file is a hierarchical structure that drives the dynamic user interface and data storage on the mobile application. The file also allows the definition of standard BLE devices and services, for example, heart rate. The primary purpose of the file is to (1) provide human-readable context about the specifics of the device and (2) provide a structure to data stream definitions to drive dynamic database table creation with efficient typing and rendering dynamic user interfaces. We provide a JSON Schema and samples on GitHub. The mobile app can reference this file from a public hosted URL, like GitHub Gist or PasteBin. We realize there is room for automation in this intermediary step, such as generating from code from the JSON object. Still, we argue that updating a configuration is an improvement over requiring mobile app updates when firmware code is changed. We plan to, at least partly, automate this step with a finalized schema as the features and specifications.

5.3 Mobile Application

The mobile application operates alongside the wearable. It does not require code updates between experiments, essentially ingesting any data sent from the device with the assistance of the specification file. We present the app as a Swift package and an iOS application. The Swift package parses the device specification file, BLE connections, and database management with SQLite. Tables for configuration and data characteristics are created dynamically for each Data Stream, along with supporting tables for storing changes to specifications, other supported sensor data, and experiment objects. Besides sensor data, other database tables aid in the development and debugging, such as storing BLE device connections, BLE data throughput, and the functionality to support the specification file changes on the fly without data loss. The intent of separating the codebases is to allow the development of additional applications utilizing the internal data structure using only the Swift Package, for example, a desktop-based command-line interface (CLI) which we are currently exploring.

The application provides a user interface for the Swift package. Users can view device information, manage BLE connections, view and export data (figures: 3b, 3c), edit data stream parameters (figure: 3a), and create experiments (figure: 3d). Timestamps index most data, mimicking a time-series database, making it easy to query and expand the UI. Sensors from the phone can also be stored. Currently, the device supports storing location data from the mobile phone's GPS component. In the future, we envision the integration of other sensors or the ability to perform additional computations, such as common data analysis or preprocessing techniques.

Experiments are second-order objects in the mobile app's database, allowing users to timebox sensor data for easy organization, export, and analysis, for example, running an experiment for a specific sensor parameterization. Experiments can also include Time Markers. Time Markers are simple records that hold a single timestamp worth noting in an experiment, for example, when an unexpected event occurs. In addition, the SQLite data is exportable, allowing for easy external analysis. Uploading data to centralized time-series databases, which can help with monitoring of long-term studies, is a planned feature.

6 DEMONSTRATION

We present data from real-world experiments using our toolkit in developing a new hardware device for exploring circadian rhythm cycle detection. In these initial experiments, we collect Photoplethysmography (PPG) and accelerometry data on a custom device based on the Nordic nRF5340 SoC running Zephyr RTOS. We collect PPG from a custom component containing an array of 9-LEDs of different wavelengths and a photodiode to experiment with blood volume detection to infer heart rate and pulse transit time, inspired by Lui et al. [9]. We collect accelerometry data from an onboard ISM330 inertial measurement unit (IMU). Additionally, we collect heart rate as a ground truth comparison using a Polar H10 chest strap and GPS location data, when in the wild, from the phone. We collect data from in-lab, stationary, and in-the-wild walking outside. All data is aggregated on the mobile app in a local SQLite database and is exported for analysis in Python to display our results. Our main goal is to test the functionality of data collection, although in the next section, we talk about initial findings. Figure 2 outlines the general routine we used for conducting experiments with our toolkit.

6.1 Methodology

We collected data from 4-participants which included authors and other lab members. In total, we collected 3 hours and 6 minutes of data (1:26:32 in the lab and 1:40:37 in the wild) in 14 experiments. For all sessions, participants wore the device on the middle forearm. In-lab participants were primarily stationary but were allowed to work at their computers. For in-wild, participants walked around campus or in a city neighborhood. Participants traveled 7.66 KM measured from location data gathered where GPS accuracy was within 10m. The main goal of developing the wearable is to test different wavelengths for heart rate indications. Therefore we experimented with two different configurations of the PPG sensor, one measuring the green and blue LEDs and the second measuring green, orange, and infrared. In addition, we altered the accelerometry sensor frequency simply for demonstration purposes, although in analysis uncovered a sensing issue in the firmware. We recorded two experiments in-lab and in-wild: first with PPG blue (460nm) and green (525nm) wavelengths, accelerometer frequency of 20Hz, and second with PPG green (525nm), orange (590nm), and infrared (940nm), and accelerometer frequency of 40Hz. 50Hz is the desired frequency for all PPG data. These parameters define the BLE specification outlined in the previous section and are updated using the

mobile application while lab members wear the device. An example segment of the data collected is shown in figure 4.

During experiments, the mobile application informs the wearer in real-time of operation, including the current state of the connected device(s) and values of current parameters. In addition, users can view sensor data or any table in the SQLite database at-a-glance to ensure the wearable is working correctly. Unfortunately, the graph displayed in figure 3c was unavailable during our experiments as it required, at the time, a beta version of iOS. Finally, we kept the iOS application active during experimentation (app in the foreground and with the screen on) to ensure the OS did not kill the process in the background, which happens periodically. We are currently exploring methods of handling BLE background processing and app termination.

6.2 Findings

This demonstration intends to show our toolkit's data collection capability in free-living situations and how it can assist in developing novel wearable devices. The benefit to the development team was the feasibility and initial testing of the PPG sensor and general device functionality. Our data revealed useful findings, confirming the successful implementation of device drivers and where further improvement is needed. Regarding usability and parameter updates, we encountered no issues during these initial trails, although we potentially attribute a few BLE device disconnections to the antenna size. The software in future versions will manage reconnection attempts.

Figure 4 shows an example 25-second in-wild sample of data collected from the wearable. Presented is PPG data with a bandpass filter (0.2Hz to 20Hz) and a rolling mean with a 5-second window. Also shown is accelerometer data with individual axes z-score normalized and an additional plot with L2 normalization. Visually, the findings are promising in confirming the functionality of the sensors. We see an expected oscillation of the accelerometry L2 normalization, presumably indicating arm swinging while walking. The filtered PPG signal for orange and green wavelengths are nearly aligned and show potential indications of heart rate with pulse onset (floor), systolic peak (first, highest peak), and diastolic peak (lower second peak). However, confirmation of the ground truth requires further analysis. We also report data reliability, which compares the expected number of records at a given sensor frequency versus the actual number recorded. The reliability of the PPG (98.72%) and especially the accelerometer (90.79%) hint at issue in the collection. Additionally, the infrared signal is weak of non-existent which requires further exploration. Although these findings highlighted in the presented 25-second sample were consistent across all experiments, accelerometer reliability is higher in experiments sensing at 20Hz, 105% versus 93% for 40Hz. PPG reliability averaged 98.4% for all collected data. With further analysis of the infrared signal, we attributed the weak signal to overpowering the LED. Since LED brightness is a configurable value for each LED in the sensor's GATT service, we were able to test this easily, confirming the issue. In previous projects, these types of issues can take significant time to identify. Similarly, we were able to identify issues with reliability in the sensor drivers.

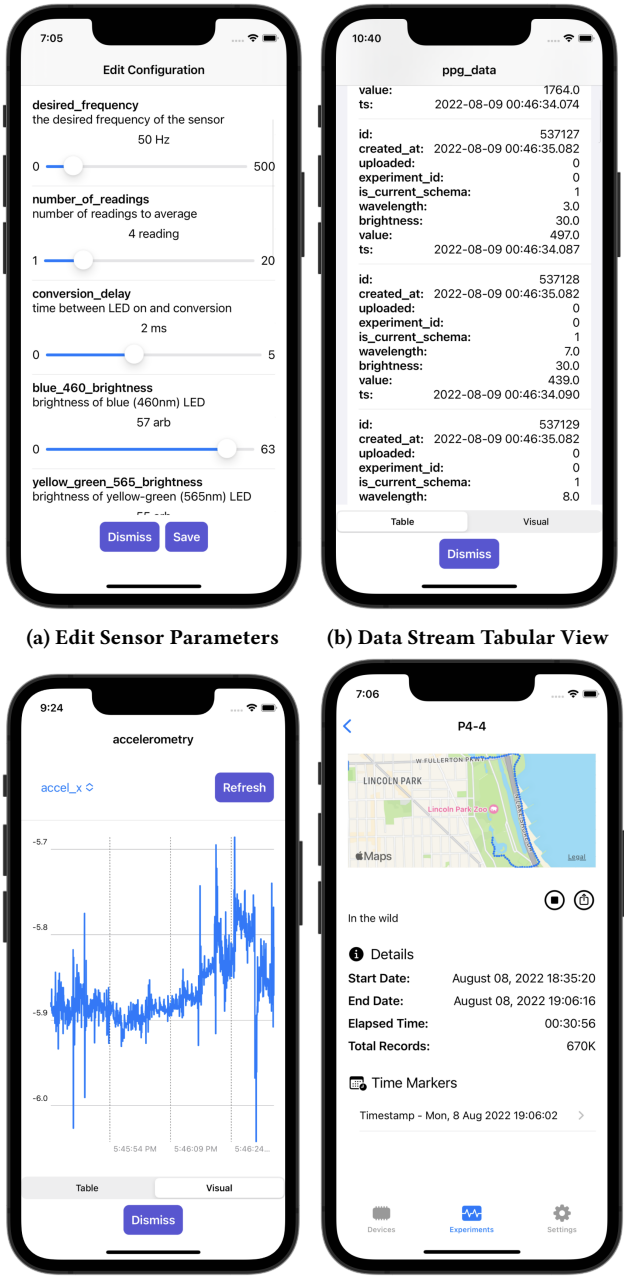
6.3 Baseline Throughput and Reliability Test

To understand the data reliability in our experiments, we conducted a baseline test to isolate the cause and test the higher throughput. This test uses a mock service on our wearable that sends randomly generated numbers to simulate a sensor and separates any potential time loss in sensor drivers. We experimented with approximately one million records with a frequency of 500Hz; this took 37 minutes and 53 seconds. In addition, the mobile application collected data in the background while the phone was used for other processes or with the screen off for segments of the experiment. The results are shown in figure 5, indicating a near-perfect data collection. This baseline proves that our toolkit is working as expected for data collection, and the reliability issues we saw in our worn experiments are related to sensor collection or task management. The tested throughput of >10KB/s is far higher than what we needed for our testing, although well below the capabilities of BLE. Testing higher frequencies would benefit from higher timestamp precision—currently, we record with millisecond precision, which is adequate for our tests. However, some use cases may require micro or nano-second precision at the expense of larger overhead in the BLE packets. These results focus on data collection and the functionality of our toolkit.

7 LIMITATIONS AND FUTURE WORK

This demonstration is a small step towards better data collection for wearable devices. We continue active development of this toolkit as part of ongoing research in wearable health, and we hope to gain community support for other research efforts in similar spaces. We present our tool as a proof-of-concept with many opportunities for improvement. First, we realize iOS and the Apple ecosystems are not particular supportive of open source or accessibility (high cost of products), although we have not written our tool with any requirements on iOS we hope to develop other mobile applications, like Android, and gateways for other, lower cost configurations. Second, the JSON specification file is currently a manual process, we hope to automate this in the future. Finally, we have evaluated data collection on only a set of researchers who are familiar with the tool or involved it its development, in the future we hope to use our toolkit in evaluation and usability studies for the contributions of new wearable devices and onboard data processing methods. Our team is actively developing new features for the toolkit, but are welcoming new collaboration to broaden utility in the wearable community. Road maps and contribution guidelines can be found on the Github organization.

Efficient and flexible data collection from the custom sensors, such as PPG, enables us to explore many applications. In Esmaelpoor et al. [4], using PPG signals, a multistage deep neural network is proposed to estimate systolic and diastolic Blood Pressures. This approach achieves decent results in feature extraction and estimation consistency. Huang et al. [8] propose MLP-BP, a deep neural network based on MLP-Mixer is proposed to estimate blood pressure (BP) from plethysmography (PPG) and electrocardiograph (ECG). In this work, processed PPG and ECG signals pass through a multi-filter to multi-channel (MFMC), then the multi-channel data is fed into the model to estimate the blood pressure. Apart from blood pressure estimation, heart rate has indications for monitoring



(a) Edit Sensor Parameters (b) Data Stream Tabular View

(c) Data Stream Graph View (d) Completed Experiment

Figure 3: The toolkit's mobile application. (a): A user interface generated from the JSON specification file shows PPG-sensor parameters to be updated. Users can alter these parameters to test different configurations without interacting with the device. (b): A view of time-series records stored in the local SQLite database for PPG data. (c): A live view of Accelerometer data using iOS's Chart API; this gives the ability to check data on-the-fly quickly. (d): A completed experiment showing high-level details, a map of the distance traveled during data collection, and time markers recorded during the experiment.

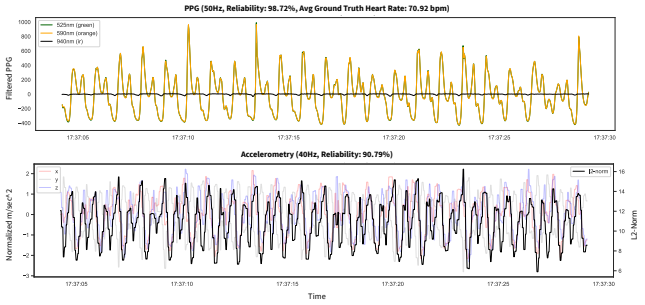


Figure 4: A 25-second example of data collected while walking. Top: PPG data with a bandpass filter (0.2Hz - 20Hz) and a 5-second rolling average from three wavelengths collected from a custom wearable. Orange and green frequencies show good alignment, but the infrared signal requires further exploration. Bottom: Accelerometer data with z-score normalization on respective axis and L2-Normalization aggregation. We report a lower than expected data reliability indicating an issue with sensor drivers. All data is exported from the mobile app and prepared using Python.

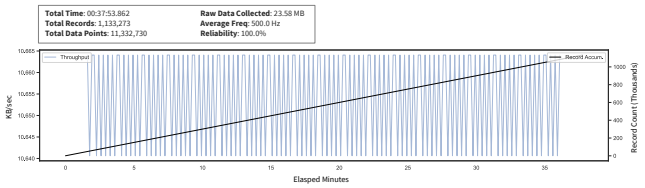


Figure 5: A demonstration data throughput test. Over 1-million records collected by a mock sensor on a nRF5240 based wearable. Displayed is a slight (3 byte) oscillation in bandwidth, which we attribute to timestamp precision, but the record count remains constant due to time-management from the wearable device. This test is well below the limits of BLE on mobile devices, and we expect we can achieve higher throughput, but would require tracking timestamps at higher precision. Data is exported from the mobile app and prepared using Python.

sleep. Radha et al. [10] propose a long short-term memory (LSTM) network to model long-term cardiac sleep architecture based on heart rate variability. This approach achieves a state-of-the-art sleep stage classification. We plan to extend our functionality to enable pipelines that support data collection and inference evaluation of real-time health metric machine learning pipelines.

8 CONCLUSION

We present the initial development of a toolkit for developing custom wearable devices in free-living scenarios. Efficient data collection of wearable devices is an engineering effort that takes time from research contributions. The toolkit's goal is to reduce the application development burden and untether early experimentation from the lab by providing a BLE guideline, a mobile application

supporting flexible bidirectional communication and adaptable data storage for wearables. We demonstrate the the toolkit's effectiveness in an active wearable project showing it's usefulness in early development cycles of new wearable hardware and presenting baseline results of data reliability. The toolkit is released open source on GitHub and is open for contributions.

ACKNOWLEDGMENTS

This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-21-2-7119. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. We thank Nikhil Khandelwal for technical assistance in the development of the iOS application and Tommy Cohen for the design of figures.

REFERENCES

[1] Brinnae Bent, Ke Wang, Emilia Grzesiak, Chentian Jiang, Yuankai Qi, Yihang Jiang, Peter Cho, Kyle Zingler, Felix Ikponmwosa Ogbeide, Arthur Zhao, Ryan Runge, Ida Sim, and Jessilyn Dunn. 2021. The digital biomarker discovery pipeline: An open-source software platform for the development of digital biomarkers using mHealth and wearables data. *Journal of Clinical and Translational Science* 5, 1 (2021), e19. https://doi.org/10.1017/cts.2020.511

[2] Wenqiang Chen, Shupei Lin, Elizabeth Thompson, and John Stankovic. 2021. SenseCollect: We Need Efficient Ways to Collect On-body Sensor-based Human Activity Data! *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (Sept. 2021), 1–27. https://doi.org/10.1145/3478119

[3] Alexander Curtiss, Blaine Rothrock, Abu Bakar, Nivedita Arora, Jason Huang, Zachary Englhardt, Aaron-Patrick Empedrado, Chixiang Wang, Saad Ahmed, Yang Zhang, Nabil Alshurafa, and Josiah Hester. 2022. FaceBit: Smart Face Masks Platform. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 4 (Dec. 2022), 151:1–151:44. https://doi.org/10.1145/3494991

[4] Jamal Esmaelpoor, Mohammad Hassan Moradi, and Abdolrahim Kadkhodamohammadi. 2020. A multistage deep neural network model for blood pressure estimation using photoplethysmogram signals. *Computers in Biology and Medicine* 120 (May 2020), 103719. https://doi.org/10.1016/j.compbiomed.2020.103719

[5] Denzil Ferreira, Vassilis Kostakos, and Anind K. Dey. 2015. AWARE: Mobile Context Instrumentation Framework. *Frontiers in ICT* 2 (April 2015), 24. https://doi.org/10.3389/fict.2015.00006

[6] Josiah Hester, Travis Peters, Tianlong Yun, Ronald Peterson, Joseph Skinner, Bhargav Golla, Kevin Storer, Steven Hearndon, Kevin Freeman, Sarah Lord, Ryan Halter, David Kotz, and Jacob Sorber. 2016. Amulet: An Energy-Efficient, Multi-Application Wearable Platform. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, Stanford CA USA, 216–229. https://doi.org/10.1145/2994551.2994554

[7] Syed Monowar Hossain, Timothy Hnat, Nazir Saleheen, Nusrat Jahan Nasrin, Joseph Noor, Bo-Jhang Ho, Tyson Condie, Mani Srivastava, and Santosh Kumar. 2017. mCerebrum: A Mobile Sensing Software Platform for Development and Validation of Digital Biomarkers and Interventions. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, Delft Netherlands, 1–14. https://doi.org/10.1145/3131672.3131694

[8] Bin Huang, Weihai Chen, Chun-Liang Lin, Chia-Feng Juang, and Jianhua Wang. 2022. MLP-BP: A novel framework for cuffless blood pressure measurement with PPG and ECG signals based on MLP-Mixer neural networks. *Biomedical Signal Processing and Control* 73 (March 2022), 103404. https://doi.org/10.1016/j.bspc.2021.103404

[9] Jing Liu, Bryan P. Yan, Yuan-Ting Zhang, Xiao-Rong Ding, Peng Su, and Ni Zhao. 2019. Multi-Wavelength Photoplethysmography Enabling Continuous Blood Pressure Measurement With Compact Wearable Electronics. *IEEE Transactions on Biomedical Engineering* 66, 6 (June 2019), 1514–1525. https://doi.org/10.1109/TBME.2018.2874957 Conference Name: IEEE Transactions on Biomedical Engineering.

[10] Mustafa Radha, Pedro Fonseca, Arnaud Moreau, Marco Ross, Andreas Cerny, Peter Anderer, Xi Long, and Ronald M. Aarts. 2019. Sleep stage classification from heart-rate variability using long short-term memory neural networks. *Scientific Reports* 9, 1 (Dec. 2019), 14149. https://doi.org/10.1038/s41598-019-49703-y

[11] Yatharth Ranjan, Zulqarnain Rashid, Callum Stewart, Pauline Conde, Mark Begale, Denny Verbeeck, Sebastian Boettcher, The Hyve, Richard Dobson, Amos Folarin, and The RADAR-CNS Consortium. 2019. RADAR-Base: Open Source Mobile Health Platform for Collecting, Monitoring, and Analyzing Data Using Sensors, Wearables, and Mobile Devices. *JMIR mHealth and uHealth* 7, 8 (Aug. 2019), e11734. https://doi.org/10.2196/11734

[12] Amanda Watson, Hyonyoung Choi, Insup Lee, and James Weimer. 2021. Raproto: an open source platform for rapid prototyping of wearable medical devices. In *Proceedings of the Workshop on Medical Cyber Physical Systems and Internet of Medical Things*. ACM, Nashville Tennessee, 1–6. https://doi.org/10.1145/3446913.3460315

[13] Shibo Zhang, Yuqi Zhao, Dzung Tri Nguyen, Runsheng Xu, Sougata Sen, Josiah Hester, and Nabil Alshurafa. 2020. NeckSense: A Multi-Sensor Necklace for Detecting Eating Activities in Free-Living Conditions. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (June 2020), 1–26. https://doi.org/10.1145/3397313