IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 12, 2019, accepted December 30, 2019, date of publication January 9, 2020, date of current version January 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965245

Continuous Blood Pressure Measurement Platform: A Wearable System Based on Multidimensional Perception Data

ZHONG DONG[1], (Member, IEEE), ZHU YIAN[1], (Member, IEEE), WANG LANQING[1], DUAN JUNHUA[1], AND HE JIAXUAN[2]

[1]School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China
[2]School of Software, Northwestern Polytechnical University, Xi'an 710129, China

Corresponding author: Zhong Dong (dzhong@nwpu.edu.cn)

This work was supported in part by the Key Research and Development Program of Shaanxi Province under Grant 2019ZDLGY03-04, in part by the Xi'an Science and Technology Plan Project under Grant GXYD19.8, and in part by the National Natural Science Foundation of China under Grant 61303225.

ABSTRACT The mobile crowd sensing technology in the environment integrating human, machines and things is an emerging direction in social computing. In kinematics research, continuous blood pressure monitoring and calibration are the basis for revealing the correlation between athlete motor function and blood pressure. At the same time, in the field of medical research, hypertension can be more easily controlled, thus improving the effectiveness of hypertension treatment. This paper presents the design principle of a human-machine fusion system based on CrowdOS, a mobile crowd sensing platform. The system innovatively establishes the correlation between blood pressure and exercise, improves the accuracy of cuffless blood pressure measurement, and verifies the feasibility of calibrating continuous cuffless blood pressure measurement based on exercise information. Using our system and electronic cuff sphygmomanometer, we measured 65 groups of data in walking, running, sitting and climbing stairs, each group lasting about 10 minutes. Based on these data, we established a regression analysis model for blood pressure measurement calibration. The accuracy of blood pressure calibration was improved from the original systolic root mean square error of 13.43mmHg and diastolic root mean square error of 8.35mmHg to 9.76mmHg and 5.56mmHg. The design method proposed in this paper provides a feasible solution for continuous cuffless blood pressure measurement and calibration, and shows broad application prospects in the fields of athlete scientific training and medical care.

INDEX TERMS IoT, human-computer interaction, blood pressure calibration, blood pressure monitoring.

I. INTRODUCTION

A. RESEARCH BACKGROUND

Blood pressure is used to measure a person health status. Hypertension is a major factor in increasing cardiovascular risk. Nearly one-third of people in the United States have high blood pressure, the number of patients in China is also increasing year by year. There are many factors affecting blood pressure, including some short-term events. Such as changes in circumstances and emotions, drug intake and exercise training of different intensity. Continuous blood pressure monitoring makes it easier for hypertension to be detected and controlled in time, which can effectively assist in the hypertension treatment [1]. In addition, in the research of kinematics, it can also provide support for the research of changes in athletes physical function during exercise. On August 26, 2014, the IEEE published the IEEE 1708 standard for wearable cuffless blood pressure measuring devices [2], based on which more wearable devices will be used to monitor blood pressure in the future [3].

Current blood pressure monitoring at home relies on an oscilloscope-based blood pressure cuff [4]. While this automatic cuff enables non-clinical monitoring without the need for a professional physician, oscillometric devices are both cumbersome and inaccurate [5], and hardware inconvenience limits the number of measurements to once or twice per

The associate editor coordinating the review of this manuscript and approving it for publication was Huaiyu Wan.

day [6]. In addition, cuff-based measurements cannot capture changes in blood pressure that occur at all points in everyday life, so incidents that affect blood pressure, such as exercise or mood changes, cannot be detected in time. Therefore, there is a need for a reliable and portable device to increase the frequency and quality of blood pressure measurement [7]. At the same time, a system is needed to construct a user's blood pressure statistical analysis model based on continuous blood pressure data and associated data of a large number of different users [8], and the model is used for automatic analysis and early warning of the user physical state. At present, the smart bracelet for measuring blood pressure has been mass-produced, but most of the smart bracelets only output sensor measurements at a certain moment when measuring blood pressure [9], and there are problems such as discontinuous monitoring, lack of statistical models, and inability to respond to physical activity. The cloud system of the bracelet does not have self-learning ability, so it is impossible to construct a blood pressure statistical analysis model based on a large amount of user data.

B. RESEARCH GOAL

We propose a scheme for measuring, analyzing and calibrating continuous cuffless blood pressure. This scheme is based on CrowdOS [10], the first crowd sensing operating system platform released by the School of Computer Science of Northwestern Polytechnical University. The platform is divided into two parts: the sensing and the server end. The sensing end is responsible for collecting sensing data such as blood pressure, heart rate and inertial measurement unit (IMU) data. After being calibrated by the regression analysis model, the collected data will be uploaded to the server end by wireless network. The server end is responsible for managing various access sensing devices, training and correcting the analysis model according to the sensing data sent by the sensing end, and further providing the analysis model trained for the user. It can also build a corresponding analysis service for the user based on the analysis model. The services developed on server end generate a RESTful API interface for the web application to obtain the information of the devices and send control instructions to the devices. Users can view the monitoring data and the working status of the devices through the web application. When the number of the system users and the historical data measured by the users reach a certain scale, the system can make decisions based on the changes in blood pressure and heart rate of different types of users according to the statistics and analysis results of big data, and give reasonable suggestions for life and exercise. Early warning will be given before the dangerous condition of abnormal blood pressure or heart rate occurs.

The rest of this paper is organized as follows: Section 2 introduces related work. Section 3 introduces the principle of system framework. Section 4 introduces the measurement and calibration mechanisms for continuous blood pressure. Section 5 gives the verification scheme for the system. The corresponding experimental results are given and we discuss and analyze the experimental results in Section 6.

II. RELATED WORK

A. WEARABLE MONITORING SYSTEMS

McArdle WD *et al.* showed that different exercise conditions have different effects on human blood pressure [11]–[14]. The Seismo Watch system developed by Carek AM et al. is a watch-type blood pressure monitor that measures blood pressure by simple operation: placing the watch against the sternum to detect micro-vibration of the chest wall associated with the heartbeat [15]. However, Seismo Watch allows the user's blood pressure to be measured at rest, but prevents continuous measurement during exercise.

The Glabella system developed by Holz *et al.* is a pair of wearable spectacles that continuously measures blood pressure by monitoring the heart rate of the wearer head [16]. Glabella can achieve continuous blood pressure monitoring, which is of great significance to us, but Glabella system measurement results are easily affected by other active noises, such as phone calls, body movements, etc.

The Naptics system developed by Carek AM and Holz C is a kind of wearable shorts that continuously monitors the wearer blood pressure without disturbing the user during sleep [17]. The Naptics system can be modified to be smaller, more integrated, and take up less space, making it possible to perform continuous and unobtrusive 24-hour blood pressure monitoring.

The Seismo system developed by Wang *et al.* is based on smart phones for blood pressure monitoring [18]. The system uses the accelerometer of a smart phone to measure vibrations caused by heart valve motion, and the camera of the smart phone measures the pulse at the fingertips [19]. The Seismo system uses a mobile phone to measure blood pressure, but it cannot monitor blood pressure continuously [20].

B. UBIQUITOUS COMPUTING SYSTEMS

This section mainly introduces several UCS(Ubiquitous Computing Systems)for different fields and scenarios [21]. They all have their own unique design perspectives and a systematic approach to solving domain problems. In the end, we summarize the commonality and uniqueness of CrowdOS compared to these UCS [22].

HomeOS is a platform that simplifies the task of managing and extending technology in the home by providing a PC-like abstraction for network devices to users and developers [23]. It uses network devices as peripherals with abstract interfaces, implements cross-device tasks through applications written for these interfaces, and provides users with a management interface designed for the home environment. HomeOS already has dozens of applications and supports a variety of devices.

CampusOS is an operating system that manages the network resources of a university campus [24]. It provides flexible support for campus application development through an

SDK that includes campus-related APIs. Developers can also easily extend the OS feature and the SDK.

Terrence et al. are developing an interaction infrastructure called the Human-Robot Interaction Operating System (HRI/OS) [25]. The HRI/OS provides a structured software framework for building human-robot teams, supporting a variety of user interfaces, enabling humans and robots to participate in task-oriented conversations and facilitating robot integration through extensible APIs.

ROS is an open source robot operating system. It relies on native OS of heterogeneous computing clusters and provides a structured communication layer on top of them. In [26], they discussed how ROS is associated with existing robotic software frameworks and provides a brief overview of several available applications that use ROS.

Additionally, Urban OS was proposed as a software platform to accelerate urban technology development and equipment deployment [27]. While, BOSS provides a set of system services to support applications deployed on distributed physical resources in large commercial buildings [28]. It can be seen from the analysis that these operating systems are designed to solve general problems in their field, and provide a comprehensive framework and a rich set of APIs. We have designed and implemented the overall architecture, important mechanisms and functional components of CrowdOS, while also taken into account the internal interaction and external callable interface, as well as the stability and scalability of the system. The blood pressure monitoring system we proposed is just one application on the CrowdOS.

III. SYSTEM ARCHETACTURE

A. CROWDOS KERNEL

The system architecture adopted in this paper is extended on the CrowdOS platform. This section mainly introduces the overall architecture of the system and the mechanism of the core modules of the system CrowdOS is designed to comprehensively address versatility.

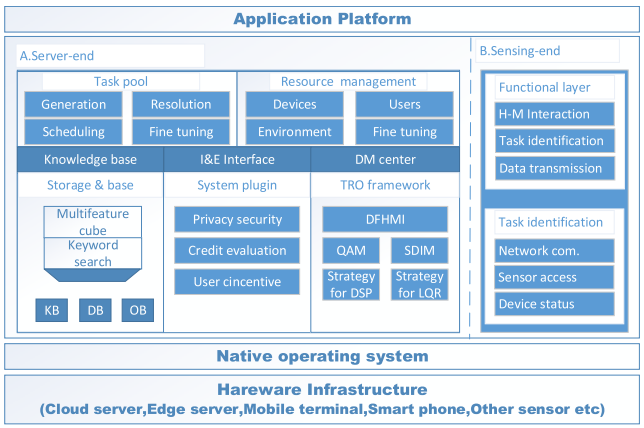


FIGURE 1. OS kernel architecture.

The OS kernel architecture is presented in figure 1 and we introduce the relationships between various modules. CrowdOS not only shields the differences of native OS running on heterogeneous devices, but also reserves interfaces of extensible function modules and personalized plugins. CrowdOS runs between the native operating system and the upper application. It includes the sensing-end and server-end. Sensing-end software consists of two types of devices. The first type is portable smart sensing devices with human-machine interaction functionalities, such as smart phones and smart watches [29]. The second type is fixed sensors deployed in the physical world which do not need to interact with people directly, such as vehicle sensors, water quality sensors, air quality sensors. In this paper, the sensing-end is further extended so that it can be deployed on Raspbian system. Server-end software provides integrated management services, which are usually deployed on server clusters, cloud servers, or edge servers. The core processing mechanisms of the OS, such as task assignment and scheduling, resource storage and management, are deployed in server end. Two ends perform data transfer and behavior control through a set of communication and interaction protocols we define.

The sensing-end is divided into two layers. The bottom layer is the system support layer, which is mainly responsible for the following functions:

- Get device status, such as current device availability, remaining power, location, etc.
- Unify packaging of sensor interfaces and data transfer formats.
- Capture available communication types and modes of device, then store them in structures.

The upper layer is the functional layer, which mainly completes three types of operations: human-machine(H-M) interaction, task identification, and data transmission.

Raw tasks can be uploaded to servers by publisher through the H-M interactive module. Participants can browse and execute tasks that have been published through smart terminals. However, for sensors without the H-M interaction module, once they are activated by the authenticated tasks, they automatically collect and upload sensor data according to predetermined rules [30]. We package the continuous blood pressure data collection module on Raspberry PI and publish it to server as a task.

Server-end combines multiple modules and innovative mechanisms. It is mainly responsible for task scheduling and assignment, resource storage, and management, data processing, and result optimization. Server-end not only handles tasks in a fine-grained manner, but also builds a unified knowledge base, while also providing a rich set of components as system plugins. Next, we briefly introduce the function of each module.

- Task pool module performs operations such as parsing, scheduling, allocation, and fine-tuning on received raw tasks.
- Resource Management module comprehensively manages the heterogeneous sensing devices, environment resources, users and task process. We package the module responsible for training the data analysis model into

task process, which is uniformly managed by Resource Management module.

- Storage and Query module provides categorized storage and rapid retrieval of massive amounts of heterogeneous data.
- System Plugin module provides a wealth of components such as privacy protection, security, credit evaluation, and user incentives.
- Task Result Optimization (TRO) framework is primarily designed to optimize results quality, which consists of deep feedback framework based on human-machine interaction (DFHMI), quality assessment mechanism (QAM), shallow-deep inference mechanism (SDIM) and specific strategies.
- Data Management Center (DMC) is mainly responsible for managing data that come with tasks, uploaded by participants, or generated during the execution process. Most of these heterogeneous multisource multimodal data are unstructured. Knowledge Base (KB) is the basis and premise of system to make reasoning. Constructing KB is an efficient way to systemically manage domain knowledge. Internal and External (I-E) interfaces include system internal interface and CrowdAPI, where the internal interface is a set of protocols used for system testing and interaction between modules. CrowdAPI provides a unified call interface for application development.

CrowdOS is designed using cloud-edge-side architecture: sensing-end is deployed on terminals to collect sensing data and special task solutions; server-end is deployed on a cloud or edge servers, which is responsible for comprehensive management of resources and real-time response to system operations; when deployed on edge servers, the OS is usually tailored and lightweight.

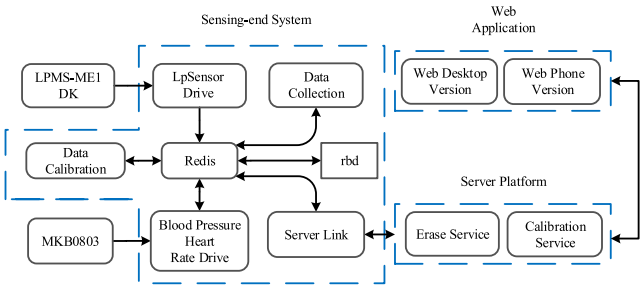


FIGURE 2. System design principle.

B. SYSTEM DESIGN PRINCIPLE

The system design principle is presented in figure 2. The software of the system is composed of sensing-end software and server-end software. The sensing-end software is responsible for reading, storing, analyzing and uploading data to the server. For the sensing-end hardware, we use Raspberry Pi 3 as the main computing device, and connect the inertial measurement module LPMS-ME1 DK through the UART serial port to collect user movement data, and connect the MKB0803 module to collect blood pressure, heart rate and wearing status data. Raspberry Pi 3 adopts Raspbian operating system. The device reads the Euler angle, quaternion, and linear acceleration data of the inertial measurement module LPMS-ME1 DK from the UART serial port through the LpSensor driver. The read data is converted into JSON format, then published to the ''imudata'' topic of the Redis process through hiredis, and waits for subscriptions from other processes. The publishing frequency is 1Hz. At the same time, the device uses the Pyserial driver to read the heart rate, systolic and diastolic blood pressure data collected by the MKB0803 sensor from the UART serial port. The driver is also responsible for sending calibration instructions to the MKB0803 sensor to reset the heart rate, systolic and diastolic blood pressure values. A clear command can also be sent to the MKB0803 sensor to erase the calibration information. The driver process converts the read data into JSON format, and publishes it to the ''blooddata'' topic of the Redis process through redis-py, waiting for subscriptions from other processes. The publishing frequency is 1Hz. The blood pressure and heart rate-driven process will subscribe to the ''property/set'' and ''property/clear'' topics of the Redis process. When calibration and clearing instructions are issued, the corresponding tasks will be executed, and the execution results will be published.

The Server Link process on the sensing-end is developed based on NodeJS. It connects to the server through the Server-iot-device-sdk library and listens to downstream messages from the server [31], including ''connect'', ''error'', and ''property/set''. After the Server Link process hears the ''property/set'' message, the program will parse it and post the analysis result to the ''property/set'' and ''property/clear'' topics via node_redis, which are used for the calibration and clearing of the MKB0803 sensor respectively. The Server Link process will subscribe to the messages of the topics ''blooddata'', ''imudata'', ''calibration'', ''erase'' and ''predblooddata''. The sensing-end system parses the messages to obtain the current working status of the device, including six states: reading, wearing error, clearing success, calibrating, calibration success and calibration failure, and report the device status, calibrated blood pressure and heart rate data to the server.

The data collection process of the sensing-end system will subscribe to the topics ''blooddata'' and ''imudata'' of the Redis process. After each startup of the sensing-end system, the data collection process will re-create a new list in Redis for blood pressure, heart rate and IMU data. The name of the list is based on time information. At the same time, the blood pressure heart rate list name and the IMU data list name are stored in the ''imu.blood.relation'' list for recording the relationship of the two lists. After the data collection process receives the data, it set timestamps on the received data and stores the data in the corresponding list.

The data calibration process will subscribe ''blooddata'' and ''imudata'' topic, and input the heart rate,blood pressure

and IMU data into regression analysis model for calibration. Calibration data can be output approximately every 10 seconds, and published in the ''predblooddata'' topic of Redis process. The application respectively uses the Bayesian Ridge regression analysis [32], the AdaBoost based Decision Tree regression analysis and Support Vector Machine regression analysis method to calibrate the heart rate [33], systolic and diastolic blood pressure data [34].

The Redis module is mainly used to solve the data storage and inter-process communication problems of the sensing-end system. If the configuration file of Redis is modified every 5 seconds, it will save the database dump.rdb to the hard disk to solve the data storage problem. Redis supports multiple data storage formats, such as keys, strings, hashes, lists and collections. The publish and subscribe function of Redis is used to solve the problem of communication between processes.

The server-end platform we developed based on CrowdOS supports multiple types of device access, and the device and server platform can perform stable and reliable two-way communication. The server platform connects various types of heterogeneous devices supporting the sensor data collection, and provides the server platform API for applications. The instruction of application is sent to the devices through API calls to achieve remote control. The sensing-end system uses the LinkJS SDK based on the NodeJS environment to access the server platform securely. The server platform has functions such as device management, data analysis, and rules engine. The Link Kit SDK based on the NodeJS environment uses the MQTT protocol for communication. The MQTT protocol is based on the TCP/IP protocol. We create two services on the server platform. The calibration service is an API call service with input parameters. The Web application calls the service by inputting the values of heart rate, diastolic and systolic pressure through the API. The erase service is an API call service without parameters. After the two services are called by the web application, the server platform will send the transmitted value to the ''property/set'' topic of the sensing-end system for calibration of the blood pressure heart rate chip. In order to meet the needs of users to view Web pages from computers and mobile phones, the system developed two versions of Web applications, desktop and mobile. The Web application of the system meets the user needs to check the working state of the equipment, monitor data in real time, detect data changes within 24 hours or calibrate the equipment.

IV. CONTINUOUS BLOOD PRESSURE MEASUREMENT MECHANISM

A. DATA ACQUISITION AND CALIBRATION MECHANISM

Due to slight posture changes or strenuous exercise, the human blood pressure and the measurement results of the wearable device could be changed. Therefore, we introduce an inertial measurement unit in the sensor module to sense the motion factor.

In order to verify the accuracy of blood pressure measurement and calibration by the system, the standard blood pressure is measured by an electronic sphygmo-manometer meeting the medical standard. At the same time, electronic sphygmomanometer and sensing-end system are used to collect data in different scenarios such as walking, running, sitting and going up and down stairs, and the data are input into different regression analysis models for comparison. The system selects regression analysis models with relatively high prediction accuracy to integrate into the sensing-end system.

During data acquisition, the wearable device can acquire continuous values including heart rate, blood pressure and motion state, and the tester can use the cuff sphygmomanometer to measure standard blood pressure for calibration of the wearable device when the wearable device is started for acquisition [35]. Ideally, the blood pressure calibration of wearable devices is a multivariable time series prediction problem.

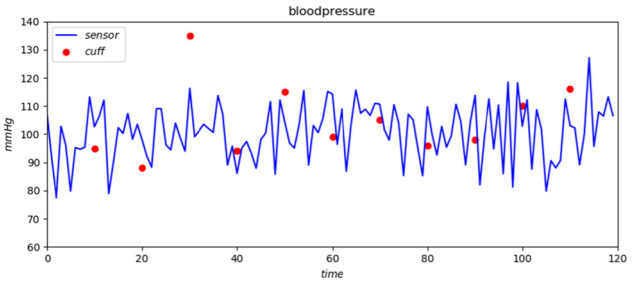


FIGURE 3. Idea data acquisition.

The collected data are shown in figure 3. the sensor label represents the blood pressure data continuously collected by the blood pressure sensor [36], and the cuff label represents the blood pressure data collected at intervals by the cuff sphygmomanometer.

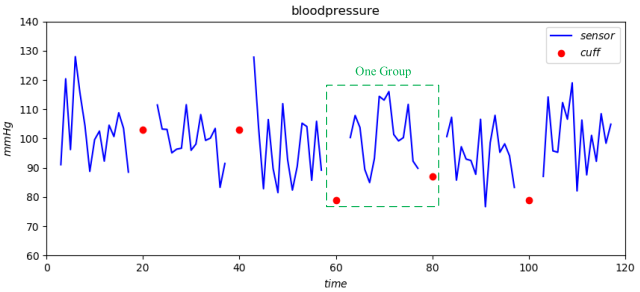


FIGURE 4. Actual data acquisition.

However, when we built the prototype system, the hardware of the sensor-end system was mainly composed of jumper wire, sensors and Rasberry Pi. The inconvenience of the hardware made it impossible for testers to measure standard blood pressure data with cuff sphygmomanometer while carrying our equipment. The actual measured data are shown in figure 4. Therefore, we use the method of extracting data in groups to collect the original data, as shown in the green boxes in figure 4. A box calibrates a group of data. The red dots on both sides represent the standard data measured by cuff

sphygmomanometer, and the middle blue curve represents the continuous blood pressure data collected by Sensing-end system for a period of time.

The first standard value measured by cuff sphygmomanometer is used to calibrate the blood pressure and heart rate sensor chip. The second standard data of the cuff sphygmomanometer and the data collected by sensing-end system are used to build a regression analysis model. During the construction of regression analysis model, the heart rate, blood pressure and movement data of sensing-end system are used to construct training vectors, and the standard data of cuff sphygmomanometer is used as training values. As shown in figure 5, the sensing-end system loads the trained regression analysis models, and constructs a prediction vector input regression analysis model by using heart rate, blood pressure and motion data in a period of time to obtain a prediction value of blood pressure at intervals.

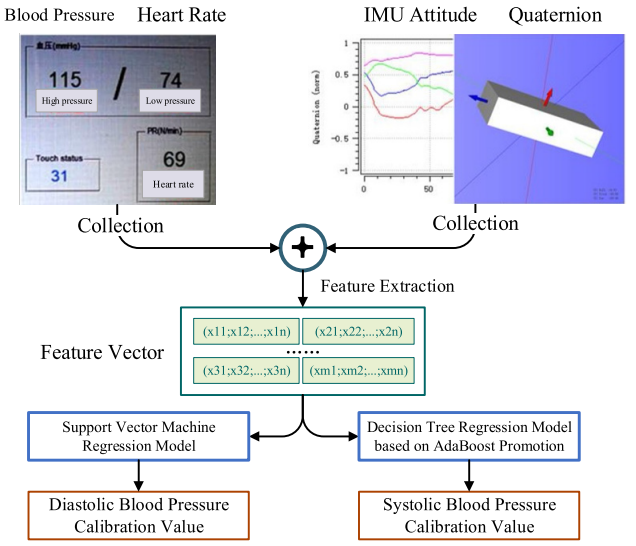


FIGURE 5. Data acquisition and calibration.

The data calibration program subscribes to the blood pressure heart rate raw data "blooddata" and the IMU raw data "imudata". Then it extracts features and structures 300-dimensional matrix from the subscription data and uses the trained Bayesian regression, the regression based on AdaBoost Improved Decision Tree and the Support Vector Machine regression model to respectively predict heart rate, systolic and diastolic blood pressure data [37].

When the data calibration program extracts features, the heart rate, systolic and diastolic blood pressure are filled into the matrix in each set of data, but the IMU data need further extraction. The original IMU data has linear acceleration, and the linear acceleration exists in three directions of the coordinate axis. The velocity can be solved according to the integral of the linear acceleration in the three directions of the defined coordinate axes, but the noise of the IMU is large in the actual work, so the integral solution is not feasible. Therefore, we calculate the absolute value of the linear acceleration of each set of data and fill it into the matrix. The calculation process of the absolute value is as follows:

$$|a| = \frac{1}{3}\sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

a_x represents the acceleration on the x-axis, a_y represents the acceleration on the y-axis and a_z represents the acceleration on the z-axis. To some extent, the absolute value of the linear acceleration $|a|$ can represent the magnitude of the change in speed. At the same time, the original IMU data also includes Euler angles and quaternions. Euler angles and quaternions represent the pose of the IMU. The Euler angle is easy to understand but generates a universal lock problem, the quaternion is easy to program and can avoid the universal lock. If each quaternion represents a pose in space, then the inverse cosine of the two quaternion dot products represents the angle between two quaternion poses. We calculate the angle between the quaternion of each set of data and the previous set of data, then fill into the matrix. The time between each set of data is about 1 second, which is equivalent to calculating the IMU bit in one second. The angle change value of the pose, the calculation process of the angle change value is as follows:

$$\Delta angle = \arccos\left(q^{pre} \cdot q^{now}\right) \times \frac{180}{\pi} \tag{2}$$

q^{pre} represents the quaternion of the previous set of data, q^{now} represents the quaternion of the current set of data, $\Delta angle$ represents the angular change of the IMU pose between the two sets of data, The combination of $|a|$ and $\Delta angle$ can represent the motion of the device. After the program normalizes the 300-dimensional matrix, enter three trained models to obtain heart rate, diastolic and systolic blood pressure prediction values, and publish the blood pressure heart rate prediction value to the "predblooddata" topic of the Redis process. The data calibration process approximately publishes forecast data every 10 seconds.

B. DATA ANALYSIS AND MODEL TRAINING

When the data collection process of the sensing-end runs, the blood pressure heart rate data collected by the MKB0803 module and the IMU data collected by the LPMS-ME1 DK module are stored in the list of the Redis process. The data collection process respectively creates a list for blood pressure heart rate data and IMU data each time the data collection process starts.

The "imu.blood.relation" list stores the relationship between the two lists. The Redis process checks the data every 5 seconds for changes. If so, saves the data in memory to disk. When Redis process starts again, loading the data on the disk can restore the state before the save completely. When the sensing-end system is offline, start the Redis process and the format conversion process. The format conversion process obtains a list of the storage relationship from the Redis process. The program traverses the storage relationship list, parses the names of blood pressure heart rate data list and the IMU data list from each storage relationship. By using

two list names, it read two lists from Redis and merges the data of the two lists. The principle of merging is that the time difference of the data in the list is less than 1 second. Finally, the format conversion process will completely convert the data stored by the Redis process into CSV format data (store in sensor.csv file), each row stores the group number, time, heart rate, systolic pressure, diastolic pressure, quaternion, Euler angle and linear acceleration.

When the data are collected, we will use the OMRON U30 to collect a set of standard blood pressure heart rate data for verification before and after collecting data from each group of devices, and store the two sets of data in the CSV format data file (cuff.csv). Each row stores the group number, time, heart rate, systolic pressure, diastolic pressure.

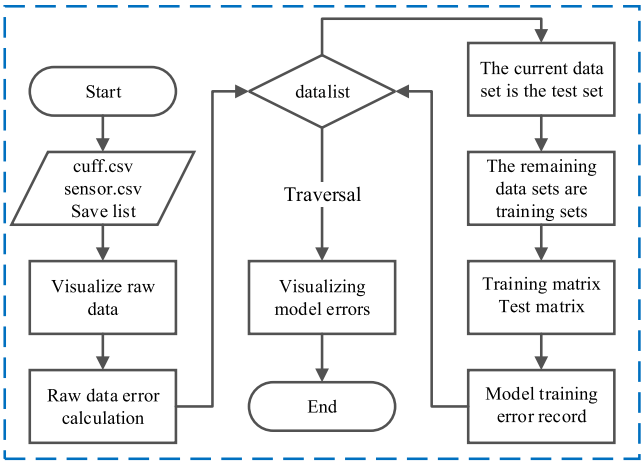


FIGURE 6. Data analysis process.

The data analysis process is shown in figure 6. When the data analysis program is initialized, the data in the cuff.csv and sensor.csv files are read and stored in a list. Each element in the list stores the same set of sensor data and standard blood pressure heart rate data. The program uses the matplotlib library to visualize the raw data stored in the list and calculate the error of the standard blood pressure heart rate data and the blood pressure heart rate data collected by the sensor. Since the time to collect one set of data by the OMRON U30 is after the sensing-end device collected the data, the time cannot be aligned. Our solution is to calculate the average error by taking the distance between the last 10 recorded values of each set of sensor data and the standard data. For example, the program uses the formula (3) when calculating the original error of a set of systolic pressures:

$$sbp_{err} = \frac{1}{10}\sum_{i=1}^{10}\left|sbp_i - sbp_{cuff}\right| \tag{3}$$

sbp_{cuff} represents the standard systolic pressure value collected by the OMRON U30 after the data is collected on the sensing-end device, sbp_i represents the i systolic pressure value of the last 10 elements of the data collected by the sensing-end device, sbp_{err} represents the systolic pressure error of the current group. The calculation of heart rate error and diastolic pressure error for each set of raw data is the same as the calculation of systolic blood pressure.

Due to the limitation of data volume, we use cross-validation method in training and verification, and traverse each group of data in turn, the current group of data is the test group, the other group of data is the training group. When constructing the training vector, the program constructs the independent variable matrix using heart rate, diastolic pressure, systolic pressure, absolute linear acceleration $|a|$ and IMU pose angle change $\Delta angle$, Use the standard blood pressure heart rate data collected by the OMRON U30 to construct a dependent variable matrix. In each round of training, the program trains Bayesian regression model, AdaBoost-based Decision Tree regression model, K-nearest Neighbor regression mode and Support Vector Machine regression model [38], calculate the prediction error of each model. When calculating the prediction error of the model, the program stores the distance between the predicted value of the model and the standard blood pressure heart rate data measured for the second time as an error. Finally, the program calculates the root mean square error of the raw data and each model, then visualizes the root mean square error. Since we have obtained raw data and each set of data errors for each model, the calculation of the root mean square error can be referred to the formula (4):

$$RMSD = \sqrt{\frac{\sum_{t=1}^{n}\left(\hat{y}_t - y_t\right)^2}{n}} \tag{4}$$

For example, when calculating the root mean square error of the original systolic pressure data, n represents the number of groups of data, t represents the group ID of the data, and $\hat{y}_t - y_t$ is replaced with the t th group systolic pressure error (sbp^t_{err}). Finally, we compare the root mean square error of the heart rate, systolic and diastolic pressures of the original data and the corresponding value calculated by each model. We select a model with relatively high precision to integrate into the sensing-end system.

V. EXPERIMENTS: DESIGN AND SETUP

The establishment of the experimental environment can be realized in two parts:

- The Sensing-end System
- The Configuration and Development of Server Platform

As shown in figure 7, the device collects systolic pressure, diastolic pressure, heart rate, quaternion and linear acceleration data through the UART serial port, then input the data into the regression analysis model to obtain the calibrated blood pressure data.

A. SENSING-END SYSTEM

The Sensing-end system device is expected to consist of the Raspberry Pi 3 Model B+ [39], the pulse sensor chip MKB0803, the inertial measurement module

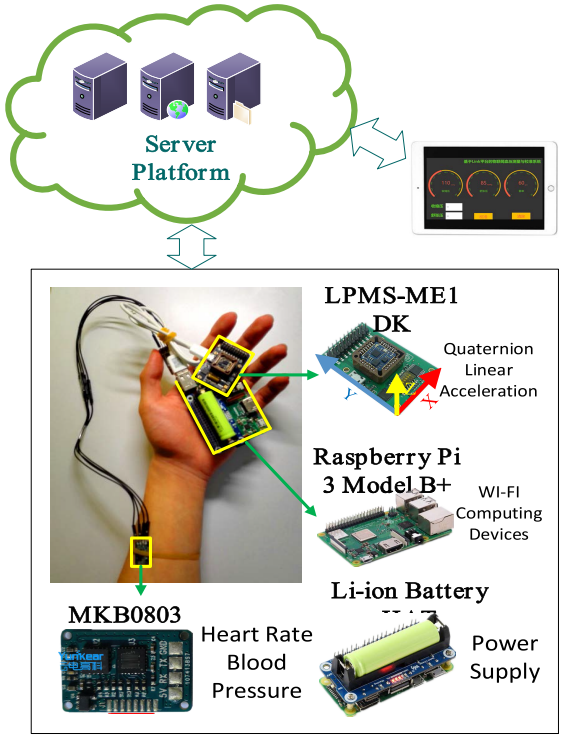


FIGURE 7. Experimental environment.

LPMS-ME1 DK and the Li-ion Battery HAT lithium battery expansion board. The MKB0803 heart rate blood pressure module is mainly composed of a YK1801 pulse sensor chip, an analog front end MN8802 pulse chip and a SFB9712 algorithm chip. The sensor collects the pulse information of the human body based on PPG [40]. After the information being processed, then output blood pressure, heart rate and other serial signals. The inertial measurement module LPMS-ME1 DK is a 9-axis inertial measurement unit that integrates sensors such as three-axis accelerometers, three-axis gyroscopes and three-axis magnetometers. It provides information such as Euler angles, quaternions and accelerations. The Li-ion Battery HAT lithium battery expansion board can be plugged into the Raspberry Pi GPIO interface for powering the Raspberry Pi. The Raspberry Pi 3 Model B+ has the support of 1.4GHz quad-core processor, dual-band 2.4GHz and 5GHz WLAN. We use the UART interface to transfer data between the Raspberry Pi 3 Model B+ and the sensor. After the Raspberry Pi 3 Model B+ caches and processes the sensor data, it uploads the information to the server platform via the WLAN.

This article uses the OMRON U30 to verify the accuracy of the device blood pressure measurement and calibration. The OMRON U30 is an upper arm type electronic sphygmo-manometer. The measurement method is oscillometric assay, and the measurement accuracy is within the range of pressure accuracy and pulse accuracy of $\pm 3mmHg$ ($\pm 0.4kPa$). The OMRON U30 complies with the AAMI standard.

TABLE 1. Definition of equipment management container.

Function name	Identifier	Type of data	Data definition
Systolic pressure	SystolicBloodPressure	int32	Ranges：1~250
Diastolic pressure	DiastolicBlood Pressure	int32	Ranges：1~250
Pulse rate	PulseRate	int32	Ranges：1~250
Erase	Erase	bool	Success -0；Clear -1；
Status	Status	enum	6 working states on the device

B. CONFIGURATION AND DEVELOPMENT OF SERVER PLATFORM

The Server Link process on the sensing-end system is connected to the server platform by using the activation certificate and the Server-iot-device-sdk library. After the access, the container can be used to report the updated attribute value to the server platform and listen to the downlink information of the server platform.

The calibration service of server platform is an API service that can be called by the Web application for the server platform to send a calibration command. The API input parameter of the calibration service is the systolic blood pressure, diastolic blood pressure and heart rate value that need to be set. The sensing-end system will transfer the API input parameter to the Server Link process. After the Server Link process listen to the calibration message of the server platform, it will execute the calibration process and report the status of the execution.

The erase service is also an API service called by the web application, which has no API entry. When invoked, the sensing-end system will send a clear command to the Server Link process. The Server Link process will perform the clear operation after listening to the clear command of the server platform and report the results to the server platform.

In server platform, we created a sensing-end device management container for the server platform system to manage remote continuous blood pressure measuring devices. The management container contains five attributes including the systolic, diastolic, pulse rate, erase and status. pulse rate attributes are integer and the specified range is from 1 to 250. The erase attribute is a Boolean type for the web application to pass the MKB0803 heart rate blood pressure module clear command. It is required that the command of clear be -1, successfully clear be -0, the status attribute is enumerated for 6 working states of the sensing-end device, which are reading -0, clearing -1, calibration success -2, calibration failure -3, calibrating -4 and wearing error -5. After the device management container is defined, we will create a new device and generate its corresponding activation certificate.

In order to meet the needs of users for real-time viewing and recording of monitoring data and working status in the experiment, we developed Web applications of server platform, including mobile phone version and desktop version. The three dashboards of the Web application are bound to the systolic pressure, diastolic pressure and pulse rate of the virtual equipment respectively. when the sensing-end system is connected to the server platform and reports calibration data, the data displayed on the dashboards will also be updated.

VI. EXPERIMENTS: RESULTS AND DISCUSSION

A. DATA COLLECTION

The data collection process of each group is as follows: first, collect the user standard heart rate, diastolic and systolic blood pressure data by using OMRON U30. Then the user wears the sensing-end system, and the system is initialized with the standard data collected for the first time. The data collection time is about 10 minutes. the system records the data of the user sitting, walking, running, and going up and down the stairs. Finally, we used the OMRON U30 to collect the user's second data. According to the data collection process, we collected a total of 65 sets of data. After the data is collected, run the format conversion program to convert the data stored in Redis to a CSV format file. The data analysis program reads the CSV file and visualizes the raw data.

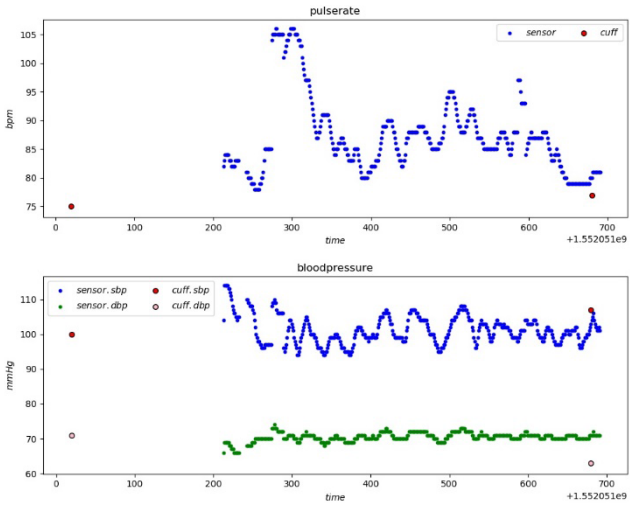
The figure 8 show the data collected by group 8. The subgraph titled pulserate in figure 8(a) represents the heart rate record value, the sensor tag records the heart rate value collected by the MKB0803 heart rate blood pressure module of the sensing-end system, and the cuff tag records the heart rate value collected by the OMRON U30. The subgraph titled blood pressure represents the blood pressure record value, the sensor.sbp and sensor.dbp tags respectively record the systolic and diastolic blood pressure values collected by the MKB0803 heart rate blood pressure module of the sensing-end system, and the cuff.sbp and cuff.dbp tags respectively record the systolic and diastolic blood pressure values collected by the OMRON U30.

The subgraph titled acceleration absolute in figure 8(b) represents the absolute value of the linear acceleration, which is calculated by the linear acceleration value acquired by the LPMS-ME1 DK at the sensing-end device and the formula (1).

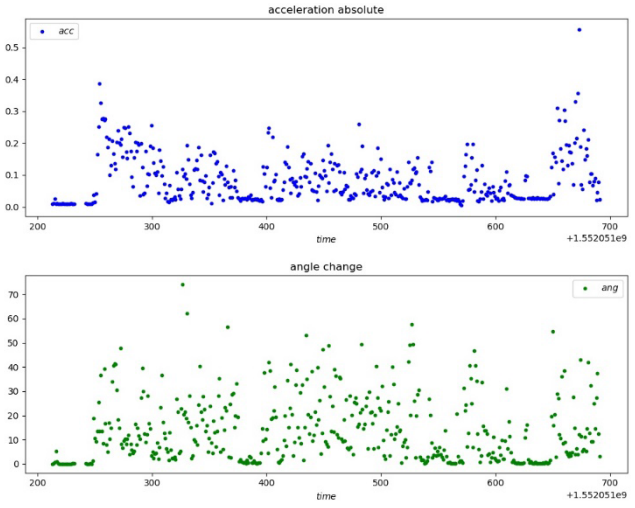
The subgraph titled angle change represents the angle change value, which is calculated by the quaternion value collected by the LPMS-ME1 DK on the sensing-end device and the formula (2).

B. EXPERIMENTS DATA ANALYSIS

After the data analysis program reads the CSV format file, the original error of each set of data is calculated according to the formula (3). The graph titled original error shows the error between diastolic and systolic blood pressure data collected by each set of devices and OMRON U30. The label systolicbp represents the original systolic pressure error, and the label diastolicbp represents the diastolic pressure original error.



(a) Blood Pressure and Heart Rate Data of Group 8



(b) IMU Data of Group 8

FIGURE 8. Data collected by group 8.

Based on the formula (4) and the original error of each set of data, the systolic root mean square error of the original data is 13.43 mmHg and the diastolic pressure root mean square error is 8.35 mmHg.

The data analysis program uses heart rate, diastolic blood pressure, systolic blood pressure, $|a|$, $\Delta angle$ data training Bayesian regression, AdaBoost-based Decision Tree regression, K-nearest Neighbor regression, and Support Vector Machine regression model, use cross-validation to verify the model. Finally, the program enters the error of each model on each set of data based on formula (4) to obtain the root mean square error of each model. The original data and the root mean square error of each model are shown in the graph titled RMSD. The decision tree regression model based on AdaBoost promotion raised the root mean square error of systolic blood pressure to 9.76 mmHg, and the Support Vector Machine regression model raised the square root error of diastolic pressure to 5.56mmHg. These two models predict

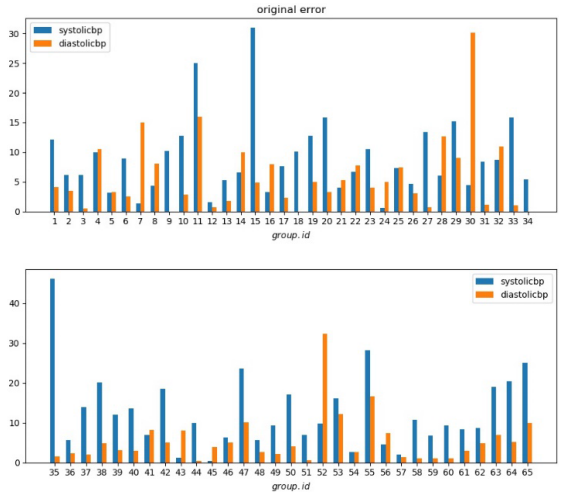


FIGURE 9. Data error of each group.

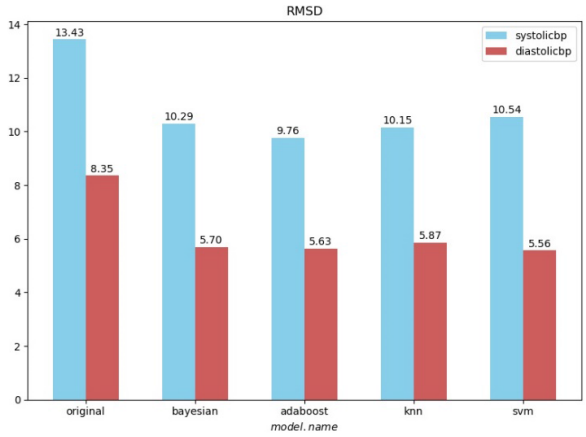


FIGURE 10. Comparison of model mean square error.

better rms error in diastolic and systolic blood pressure, respectively. Therefore, we integrate the two models into the sensing-end system. The regression model training requires multiple adjustments to achieve the best results. The Decision Tree regression model based on AdaBoost promotion performs better in systolic blood pressure prediction with a maximum depth of 5 for the tree and a maximum number of estimators for termination promotion of 900.

The Support Vector Machine regression model and the grid search hyperparameter optimization method combine the options of setting the penalty coefficient C to 1, 10, 100, and 1000. The options of the kernel coefficient gamma are 10^{-2}, 10^{-1}, 10^{0}, 10^{1} and 10^{2}, and the predicting result of diastolic blood pressure is better when the kernel function is RBF.

VII. CONCLUSION AND FUTURE WORK

A. WORK SUMMARY

This article builds the blood pressure measurement and calibration system based on the CrowdOS platform. The system can continuously collect the wearer blood pressure and use the regression model to calibrate the collected blood pressure. The wearer can view the calibrated blood pressure value through the web page and manage devices. The system consists of Sensing-end and Server-end. Sensing-end is responsible for collecting heart rate, blood pressure, quaternion and linear acceleration data and inputting the collected data into the trained regression model to obtain the calibrated blood pressure value. The sensing-end system accesses the server platform to report calibration data and receives control commands from the platform.

After the system is built and tested, we collects 65 sets of data under the condition of sitting, walking, running and going up and down stairs based on our system and OMRON U30 electronic cuff sphygmomanometer (pressure accuracy: ±3mmHg), each group of data is about 10 minutes long. The server data analysis program uses 65 sets of data to train the regression analysis model by cross-validation method. Based on the AdaBoost-enhanced Decision Tree regression model, the systolic pressure root mean square error of the original data is raised from 13.43mmHg to 9.76 mmHg, and the Support Vector Machine regression model raises the diastolic pressure root mean square error of the original data from 8.35mmHg to 5.56 mmHg.

B. FUTURE WORK

The system in this article innovatively combines continuous blood pressure monitoring with exercise data to improve the accuracy of blood pressure measurements. At this stage, the commercialized smart bracelet is still unable to continuously monitor blood pressure values, but our system can continuously measure and calibrate. However, there is still much work to be done on our research:

(1) In terms of system construction, since the Raspberry Pi 3 Model B+ is a relatively high-performance edge computing device, and the Li-ion Battery HAT lithium battery expansion board uses 5V lithium battery, the running time of the whole system is about 20 minutes, so the time is short. At the same time, the Raspberry Pi 3 Model B+ and the sensor are relatively large in size, which is inconvenient for the wearer to collect data. The system upgrade and retrofit can focus on miniaturization, convenience and long life in next phase.

(2) In terms of experimental setup, each set of collected data is isolated due to the limitation of system running time. The sensing-end hardware system is too large for our device and OMRON U30 to measure data at the same time, only to force the division of time series data into a set of 60 elements. In the future, with the improvement of the prototype system, the data acquisition method can be further improved.

(3) In terms of analysis model, regression analysis model has not yet involved time and scene information. In the next stage, long-term and short-term memory networks can be considered to reestablish the models between motion information, time information, scene information and blood pressure data. Therefore, how to combine time and scene information to improve blood pressure prediction accuracy and user behavior reasoning ability can still be further developed.

In summary, continuous and accurate blood pressure monitoring is of great significance in the fields of disease prevention and sports training. The system discussed in this article has the significance of further research and wide commercial application value.

REFERENCES

[1] E. Altintas, K. Takoh, Y. Ohno, K. Abe, T. Akagawa, T. Ariyama, M. Kubo, K. Tsuda, and O. Tochikubo, "Wearable and low-stress ambulatory blood pressure monitoring technology for hypertension diagnosis," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 4962–4965.

[2] *IEEE Standard for Wearable Cuffless Blood Pressure Measuring Devices*, Standard IEEE P1708/D04, June 2014, pp. 1–32.

[3] R. S. H. Andrew, "Non-invasive continuous blood pressure monitoring systems: Current and proposed technology issues and challenges," Australas. Phys. Eng. Sci. Med., Washington, DC, USA, Tech. Rep., Nov. 2019.

[4] Y. Imai, K. Asayama, S. Fujiwara, K. Saito, H. Sato, T. Haga, M. Satoh, T. Murakami, H. Metoki, M. Kikuya, T. Obara, R. Inoue, and T. Ohkubo, "Development and evaluation of a home nocturnal blood pressure monitoring system using a wrist-cuff device," *Blood Pressure Monitor.*, vol. 23, no. 6, pp. 318–326, Dec. 2018.

[5] P. A. Ogink, J. M. De Jong, M. Koeneman, M. Weenk, L. J. Engelen, H. Van Goor, T. H. Van De Belt, and S. J. Bredie, "Feasibility of a new cuffless device for ambulatory blood pressure measurement in patients with hypertension: Mixed methods study," *J. Med. Internet Res.*, vol. 21, no. 6, Apr. 2019, Art. no. e11164.

[6] J. Liu, B. P. Yan, Y.-T. Zhang, X.-R. Ding, P. Su, and N. Zhao, "Multi-wavelength photoplethysmography enabling continuous blood pressure measurement with compact wearable electronics," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 6, pp. 1514–1525, Jun. 2019.

[7] T. Zhang, B. Li, C. Xu, and J. Wang, "Research on general wearable technology based on embedded operating system," in *Proc. Int. Conf. Cyber Secur. Intell. Anal.*, 2020, pp. 1200–1204.

[8] V. P. Rachim and W.-Y. Chung, "Multimodal wrist biosensor for wearable cuff-less blood pressure monitoring system," *Sci. Rep.* vol. 9, no. 1, May 2019, Art. no. 7947.

[9] B. W. An, "Smart sensor systems for wearable electronic devices," *Polymers*, vol. 9, no. 8, 2017, Art. no. 303.

[10] [Online]. Available: https://www.crowdos.cn/

[11] W. D. McArdle, F. T. Katch, and V. L. Katch, *Essentials of Exercise Physiology*. 3rd ed. 2006, pp. 270–335.

[12] V. A. Cornelissen and A. Neil Smart, "Exercise training for blood pressure: A systematic review and meta-analysis," *J. Amer. Heart Assoc.*, vol. 2, no. 1, 2013, Art. no. e004473.

[13] J. Pineda-Gutierrez, L. Miro-Amarante, M. Hernandez-Velazquez, F. Sivianes-Castillo, and M. Dominguez-Morales, "Designing a wearable device for step analyzing," in *Proc. IEEE 32nd Int. Symp. Comput.-Based Med. Syst. (CBMS)*, Cordoba, Spain, Jun. 2019, pp. 259–262.

[14] Q. Xin and J. Wu, "A novel wearable device for continuous, non-invasion blood pressure measurement," *Comput. Biol. Chem.*, vol. 69, pp. 134–137, Aug. 2017.

[15] A. M. Carek, J. Conant, A. Joshi, H. Kang, and O. T. Inan, "SeismoWatch: Wearable cuffless blood pressure monitoring using pulse transit time," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–16, Sep. 2017.

[16] C. Holz and E. J. Wang, "Glabella: Continuously sensing blood pressure behavior using an unobtrusive wearable device," *ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–23, Sep. 2017.

[17] A. Carek and C. Holz, "Naptics: Convenient and continuous blood pressure monitoring during sleep," *ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 1–22, Sep. 2018.

[18] E. J. Wang, J. Zhu, M. Jain, T.-J. Lee, E. Saba, L. Nachman, and S. N. Patel, "Seismo: Blood pressure monitoring using built-in smartphone accelerometer and camera," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, Apr. 2018, p. 425.

[19] K. Liu, C. Mu, C. Wang, and Y. Cheng, "Design of wearable system for hand function monitoring," in *Proc. 3rd Int. Forum Energy, Environ. Sci. Mater. (IFEESM)*, Feb. 2018.

[20] F. Miao, Y. Cheng, Y. He, Q. He, and Y. Li, "A wearable context-aware ecg monitoring system integrated with built-in kinematic sensors of the smartphone," *Sensors*, vol. 15, no. 5, pp. 11465–11484, May 2015.

[21] H. Mei and Y. Guo, "Toward ubiquitous operating systems: A software-defined perspective," *Computer*, vol. 51, no. 1, pp. 50–56, Jan. 2018.

[22] A. P. Kesavan, N. Prakasam, A. Hegde, and B. Lagesse, "Enabling crowd sensing for non-experts," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, Mar. 2018, pp. 442–444.

[23] H. Mei and Y. Guo, "Operating systems for Internetware: Challenges and future directions," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, Jul. 2018, pp. 1377–1384.

[24] P. Yuan, Y. Guo, and X. Chen, "Towards an operating system for the campus," in *Proc. 5th Asia–Pacific Symp. Internetware*, 2013, Art. no. 24.

[25] M. Mohan and K. J. Kuchenbecker, "A design tool for therapeutic social-physical human-robot interactions," in *Proc. 14th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2019, Daegu, Korea, Mar. 2019, pp. 727–729.

[26] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Sour. Softw.* Kobe, Japan, vol. 3, May 2009, p. 5.

[27] L. P. SA. *The Urban Operating System*. Accessed: Jul. 30, 2019. [Online]. Available: http://living-planit.com/

[28] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building operating system services," in *Proc. Symp. Netw. Syst. Des. Implement. (NSDI)*, 2013, pp. 443–457.

[29] M. Kuwabara, K. Harada, Y. Hishiki, and K. Kario, "Validation of two watch-type wearable blood pressure monitors according to the ANSI/AAMI/ISO81060-2:2013 guidelines: Omron HEM-6410T-ZM and HEM-6410T-ZL," *J. Clin. Hypertension*, vol. 21, no. 6, pp. 853–858, Jun. 2019.

[30] S. Din and A. Paul, "Smart health monitoring and management system: Toward autonomous wearable sensing for Internet of Things using big data analytics," *Future Gener. Comput. Syst.*, vol. 91, pp. 611–619, Feb. 2019.

[31] F. Lacerda, M. Lima-Marques, and A. Resmini, "An information architecture framework for the Internet of Things," *Philos. Technol.*, vol. 32, no. 4, pp. 727–744, Dec. 2019.

[32] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.* vol. 4, no. 3, pp. 415–447, 1992.

[33] H. Drucker, "Improving regressors using boosting techniques," in *Proc. ICML*, vol. 97, Jul. 1997, pp. 107–115.

[34] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *TISTACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[35] B. M. G. Rosa and G. Z. Yang, "A flexible wearable device for measurement of cardiac, electrodermal, and motion parameters in mental healthcare applications," *IEEE J. Biomed. Health Inform.*, vol. 23, no. 6, pp. 2276–2285, Nov. 2019.

[36] J. Jiang, J. Xu, H. Zhou, and Z. Yan, "Wearable device for non-invasive continuously blood pressure monitoring," *Chin. J. Med. Instrum.*, vol. 42, pp. 400–404, Nov. 2018.

[37] E. Lee, Y.-D. Seo, and Y.-G. Kim, "Self-adaptive framework based on MAPE loop for Internet of Things," *Sensors*, vol. 19, no. 13, p. 2996, Jul. 2019.

[38] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Stat.*, vol. 46, no. 3, pp. 175–185, Aug. 1992.

[39] B. Mohebali, "A scalable communication abstraction framework for Internet of Things applications using raspberry pi," *Proc. SPIE Disruptive Technol. Inf. Sci.*, vol. 10652, May 2018, Art. no. 1065205.

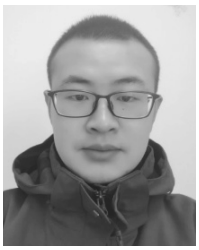
[40] L. Meili and L. Zhen, "Real-time monitoring system of human physiological parameters based on SON308," *Techn. Automat. Appl.* vol. 12, p. 37, 2018.

ZHONG DONG (Member, IEEE) received the B.Eng. degree in information security, the M.Eng. degree in software engineering, and the Ph.D. degree in computer science and technology engineering from Northwestern Polytechnic University, China, in 2002, 2005, and 2010, respectively, where he is currently an Associate Professor with the School of Computer Science. His research interests include machine learning, artificial intelligence, and the Internet of Things. He is a member of the ACM and CCF.

ZHU YIAN (Member, IEEE) received the Ph.D. degree in computer science and technology from Northwestern Polytechnic University, China, in 1994, where he is currently a Professor. He has published more than 80 articles. His research interests include mobile computing, parallel computing, embedded systems, aeronautical ad hoc networks, and the Internet of Things. He is a member of the ACM and CCF.

WANG LANQING received the bachelor's degree in computer science and technology from Northwest University, in June 2019. She is currently pursuing the degree with the School of Computer Science, Northwestern Polytechnic University, China. Her research interests include machine learning and computer system architecture.

DUAN JUNHUA received the B.Eng. degree in mathematics and computer science from Shanxi Normal University, in 2002, and the M.Eng. and Ph.D. degrees in computer science and technology from Northwestern Polytechnic University, China, in 2005 and 2015, respectively, China. She is currently a Lecturer with the School of Computer Science, Northwestern Polytechnic University. Her research interests include machine learning, artificial intelligence, and embedded systems.



HE JIAXUAN received the bachelor's degree in microelectronics from Northwestern Polytechnic University, China, in July 2018, where he is currently pursuing the master's degree with the Department of Software Engineering. His research interests include artificial intelligence and machine learning.