nature protocols

https://doi.org/10.1038/s41596-025-01296-9

Protocol

Check for updates

Deployment of a cloud-based passive defecation monitoring system for continuous gut health monitoring

A list of authors and their affiliations appears at the end of the paper

Abstract

With the growing demand for accurate yet effortless health monitoring at home, most current approaches to stool analysis rely on self-reported diaries that are prone to recall bias and low adherence. Here we present a fully passive alternative: the Precision Health Integrated Diagnostic (PHIND) system, a smart-toilet-based platform that enables automated defecation monitoring without requiring users to alter their daily routines. By integrating optical and pressure sensors with cloud-based convolutional neural networks, the PHIND system classifies stool form according to the Bristol Stool Form Scale and records key defecatory parameters, including total event time, defecation duration and time to first stool drop. The protocol proceeds in three principal stages: (1) assembling and mounting the hardware onto a conventional toilet; (2) training convolutional neural network models for stool classification and event detection; and (3) image acquisition and deploying cloud infrastructure for real-time analysis, data storage and visualization. Compared with traditional methods that depend on user-reported stool diaries, PHIND provides objective, near real-time data free from recall error, enabling more reliable early detection and long-term management of gastrointestinal conditions. Researchers and clinicians can expect high classification accuracy and robust, longitudinal insights into defecation patterns. The complete protocol—from hardware setup to system validation—can typically be completed within 2 d, excluding printed circuit board manufacturing, which generally requires up to 15 d depending on the manufacturing provider.

Key points

- This smart-toilet-based platform for automated defecation monitoring integrates optical and pressure sensors with cloud-based convolutional neural networks to classify stool form and record key defecatory parameters, including total event time, defecation duration and time to first stool drop.
- Compared with traditional methods that depend on user-reported stool diaries, it provides objective, near real-time data free from recall error, enabling more reliable early detection and long-term management of gastrointestinal conditions.

Key references

Park, S.-m. et al. *Nat. Biomed. Eng.* **4**, 624–635 (2020): https://doi.org/10.1038/s41551-020-0534-9

Ge, T. J. et al. *npj Digit. Med.* **5**, 39 (2022): https://doi.org/10.1038/s41746-022-00582-0

Ge, T. J. et al. *Sci. Transl. Med.* **15**, eabk3489 (2023): https://doi.org/10.1126/scitranslmed.abk3489

Park, S.-m. et al. *Nat. Rev. Gastroenterol. Hepatol.* **18**, 521–522 (2021): https://doi.org/10.1038/s41575-021-00462-0

Song, Z. et al. *Adv. Sci.* **12**, 2503247 (2025): https://doi.org/10.1002/advs.202503247

e-mail: brianjlee@skku.edu; park.seungmin@ntu.edu.sg

Introduction

Continuous health monitoring is becoming increasingly important in the modern landscape of personalized medicine and precision health[1–3]. Smart home environments now offer an opportunity to leverage everyday devices for health surveillance, with the toilet offering a unique platform for noninvasive health monitoring[4]. Among various physiological indicators, stool serves as a valuable biomaterial that reflects an individual's gut health and overall well-being[5,6]. Stool characteristics such as its form[7], consistency, color and frequency are key indicators of gastrointestinal health and broader health conditions, ranging from nutritional deficiencies to severe conditions such as colorectal cancer and inflammatory bowel disease (IBD)[8,9].

Traditionally, stool observations have been collected through stool diaries or self-report questionnaires, which can be prone to recall bias and inaccuracies (https://www.niddk.nih.gov/-/media/Files/Weight-Management/Stool_Diary_508.pdf)[9,10]. Many newer approaches, including smartphone applications, still require active user participation and manual logging. For instance, apps such as Poop Tracker simply digitize the classic stool diary, demanding that users rate and record each bowel movement. Other applications, such as Dieta and Auggi, sought to implement artificial intelligence (AI) for Bristol Stool Form Scale (BSFS) classification but still required users to capture and upload images of their stool, an additional step that imposes lifestyle changes and may discourage long-term adherence. Notably, these AI-based apps are no longer publicly available, presumably because they relied on manual input rather than fully automated recording[11–13]. This highlighting a broader challenge among digital stool-monitoring solutions: low adherence rates when users are required to change their daily routines[4].

In this work, we introduce the Precision Health Integrated Diagnostic (PHIND) system, a fully passive, near real-time defecation monitoring platform designed to address these limitations[14]. By integrating optical and pressure sensors with machine learning algorithms, our system automatically collects data on each defecation event without requiring any active input or behavioral adjustments from the user. Convolutional neural networks (CNNs) trained on medically annotated datasets classify stool according to the BSFS, providing information on total event time, defecation duration and first stool dropping time. These measurements, when collected continuously, can aid in the early detection and monitoring of gastrointestinal conditions. Although the PHIND system focuses on stool form and timing parameters, offering only a partial snapshot of gut health, its ability to capture consistent, objective data may still be advantageous—particularly for individuals seeking to track changes in daily bowel patterns or investigate potential triggers. It should be noted that inflammatory markers, microbial composition and other in-depth analyses remain beyond the scope of this system, limiting its direct applicability to disorders with highly variable or multifactorial presentations, such as irritable bowel syndrome (IBS) and IBD.

To promote accessibility and reduce barriers to adoption, we provide the complete codebase for both model training and system control, enabling nonspecialists to deploy and modify the PHIND system. We have also modularized the control code to make it easy to add new features in the future if users or clinicians wish to extend its capabilities. Detailed hardware installation instructions are provided, similar to installing a typical bidet seat. Most setups can be completed in ~5 min. We carefully optimized the system to keep it lightweight, ensuring accurate results while lowering cloud computing costs (US$0.0146–0.196 per hour, depending on use) and keeping hardware costs ~US$100. This makes the system affordable for both personal use and larger-scale deployment in institutions.

The PHIND system is particularly suitable for households that require daily stool monitoring but are unwilling or unable to engage in time-consuming manual recording and who seek a more accurate way to track defecation patterns. Although various tools—both electronic and paper-based—are currently available for stool recording, there is still no system on the market that can monitor defecation conditions without requiring any change to a person's daily routine. In this protocol, we describe how to build such a system, which may serve as an early warning tool for gut health. In addition, the PHIND system is not a commercially available

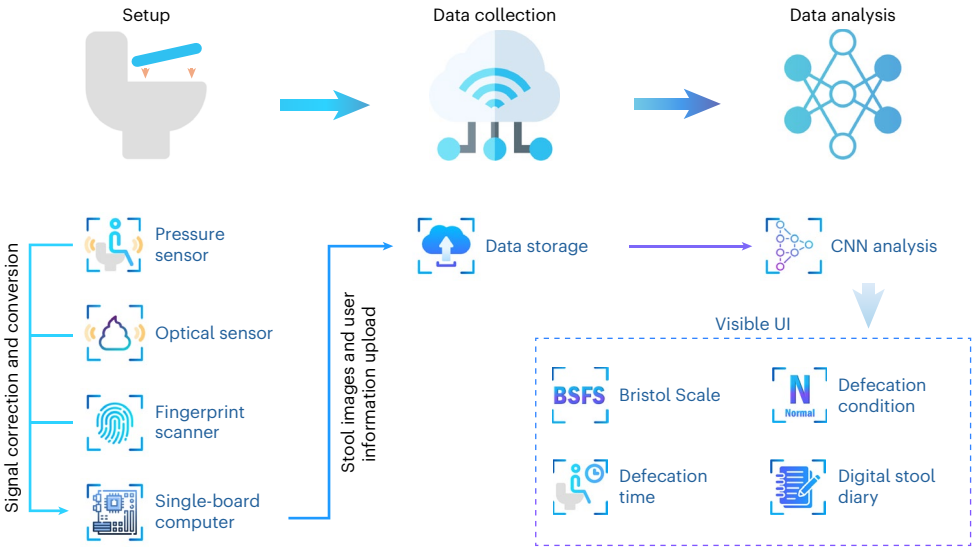

Fig. 1 | The PHIND system. A pressure sensor, optical sensor, fingerprint scanner and a single-board computer are mounted onto a sitting toilet system. These components capture analog signals from user activity, which are then converted into digital signals. The single-board computer uploads the data into the cloud, where they are processed and analyzed by the CNN models on the cloud. The resulting information is displayed through a UI to include information on BSFS classification, defecation condition, defecation time and a longitudinal digital stool diary.

off-the-shelf product; users cannot simply purchase and use it directly at home. Instead, they must assemble the system themselves by following the step-by-step instructions provided in this protocol.

Development of the protocol

The inception of our protocol emerged from recognizing the need for integrated, long-term health monitoring technologies, focusing on defecation analysis for the generation of actionable, longitudinal biometric data. We developed the PHIND system, an autonomous healthcare device advanced with deep learning techniques, which demonstrates accuracy on par with trained medical professionals[4].

Our protocol represents a novel advancement by drawing inspiration from continuous monitoring methodologies, such as digital twins used in aircraft engine maintenance to prevent failures. We adopted a passive health monitoring approach[15] to focus on disease prevention and early detection[1–3]. Unlike wearable devices that require active user engagement, our toilet-based system can operate passively, ensuring long-term user compliance by minimizing user input. It also requires only a basic electricity supply to operate, which makes it sustainable with minimal maintenance and eliminates the need for disposable tools such as test strips. Ultimately, we hope this protocol has the potential to transform personal healthcare by opening new avenues for integrating real-time biometric data into broader healthcare frameworks, contributing to more personalized data-driven clinical decisions.

As shown in Fig. 1, the defecation process is monitored by a single-board computer equipped with an optical sensor and a pressure sensor. The optical sensor captures stool images during defecation, whereas the pressure sensor detects user sitting, triggering the system to initiate data collection. A fingerprint scanner is activated during flushing, ensuring accurate real-time user identification. All data are automatically and securely transmitted to a cloud server, which is processed through machine learning models trained on medically annotated datasets.

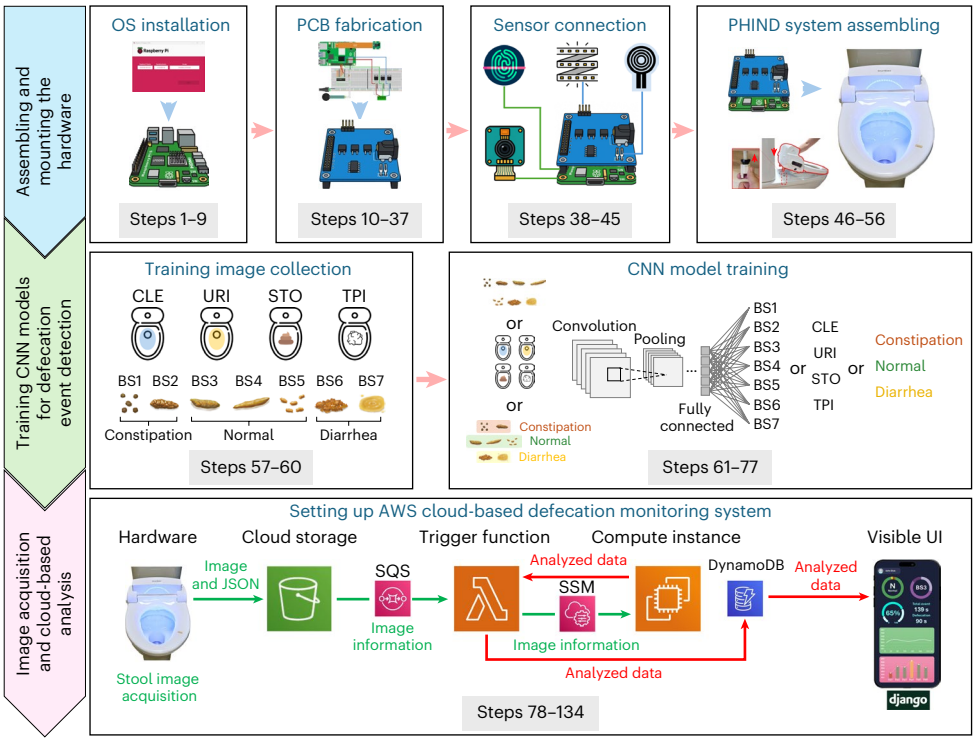

Fig. 2 | Workflow showing the preparation and deployment of the PHIND system. The deployment of the PHIND system involves three main processes: (1) assembling and mounting the hardware; (2) training the CNN models for defecation event detection; and (3) image acquisition and cloud-based analysis. The full procedure comprises a total of 134 steps. CLE, clean; URI, containing urine only; STO, containing stool (urine may exist); TPI, containing toilet paper.

In addition to recording defecation conditions and BSFS classifications, the system also records meta-information such as defecation time and user identification, integrating these into a longitudinal digital stool diary. This provides a comprehensive user profile through a visual user interface (UI). The system implements backend analytics in a cloud system to receive and interpret raw data, and we have developed a front-end interface (web application) for users to access their results. We utilized various modules in a cloud service (in this protocol, Amazon Web Services (AWS)) such as Simple Storage Service (S3) buckets, Lambda functions, Elastic Compute Cloud (EC2) instances and DynamoDB to build a scalable and efficient infrastructure. The detailed flow chart, including the step numbers, is shown in Fig. 2.

The system supports personalized healthcare management with secure data handling and potential integration with electronic health records systems. In practice, this system must adhere to privacy and security protocols such as the General Data Protection Regulation or the Health Insurance Portability and Accountability Act.

To ensure compliance with the General Data Protection Regulation and the Health Insurance Portability and Accountability Act, the proposed protocols incorporate comprehensive privacy-by-design principles at multiple stages. At the local data-collection level, biometric authentication via fingerprint matching is exclusively conducted on device, preventing the transmission or storage of sensitive personal identifiers in the cloud. Captured metadata, such as user identifiers, timestamps and event durations, are pseudonymized before cloud processing, thereby minimizing personal data exposure. The data transmissions from the local devices to the cloud infrastructure utilize encrypted communication channels through Hypertext Transfer Protocol Secure/Transport Layer Security protocols to maintain confidentiality and integrity during transit. Upon reaching cloud services, the data are securely stored using server-side encryption with AWS Key Management Service, ensuring robust cryptographic protection at rest.

Moreover, the system employs stringent role-based access controls managed via AWS Identity and Access Management (IAM), ensuring that only authorized services and users can access data within strictly defined parameters. The real-time analysis pipeline leverages event-driven AWS Lambda functions, and it is recommended that these functions be configured with the minimal necessary permissions to significantly reduce potential security vulnerabilities. The auditability and transparency are further reinforced through logging mechanisms integrated into AWS services such as DynamoDB and S3, enabling traceable and accountable data handling practices. Users are required to assemble the system according to the documented protocols and to ensure that their AWS account is properly secured to prevent any unauthorized access or compromise of personal privacy.

Overview of the procedure

The protocol proceeds in three principal stages: (1) assembling and mounting the hardware onto a conventional toilet (Steps 1–56), (2) training CNN models for stool classification and event detection (Steps 57–121) and (3) image acquisition and deploying cloud infrastructure for real-time analysis, data storage and visualization (Steps 122–132).

Applications

The PHIND smart toilet system enables the continuous, passive monitoring of gut health within the user's private environment. Although it may not provide a definitive diagnosis for complex conditions such as IBD or IBS, it can detect valuable early signals by capturing changes in stool form and defecation patterns over time. These objective, longitudinal data may help identify symptom fluctuations or potential flare-ups, thereby supporting timely medical evaluation and potentially contributing to the long-term management and screening of IBD and IBS.

Employing machine learning algorithms for stool classification and parameter analysis offers a noninvasive complementary test to traditional methods such as colonoscopy and stool sample analysis, which can be uncomfortable and inconvenient and require active patient participation. Continuous data collection affords a longitudinal view of a patient's gut health, enabling healthcare providers to track disease progression or response to treatment in real time. In clinical settings, the PHIND system can assist in personalized treatment regimens, especially for patients undergoing therapies that impact gut health, such as chemotherapy[16] that often causes mucositis and enterocolitis, by providing ongoing monitoring to swiftly identify and address adverse effects.

In addition to its clinical applications, the PHIND system can replace current subjective and cumbersome stool diaries, often used for managing various gastrointestinal symptoms. By automating this process, the system provides more accurate and consistent data, enhancing the management of gastrointestinal symptoms and reducing the burden on patients to record their stool characteristics manually.

Beyond individual healthcare, the system's utility extends to population health management. Given the PHIND system's features, such as being easily mountable on existing toilets and enabling passive monitoring, it lends itself to large-scale deployment and continuous gut health monitoring across a population[3,4]. Aggregated data can reveal trends that inform public health initiatives and preventative strategies[17,18]. Robust data collection can contribute to research on the gut microbiome as stool consistency and gut transit time are major factors that affect its composition[19,20].

The system's capability for user identification ensures accurate data recording, making it suitable for multi-user environments such as nursing homes or assisted living facilities. By maintaining individual health profiles, the system can assist caregivers in managing the health of multiple residents efficiently and effectively.

Advantages and limitations

Advantages

The PHIND system transforms traditional stool diaries[7,21–23] into a digital format[4,24], providing an objective and automated monitoring of defecation events[25–27], which are valuable digital biomarkers for clinical interpretation. By overcoming the inaccuracies of patient-reported

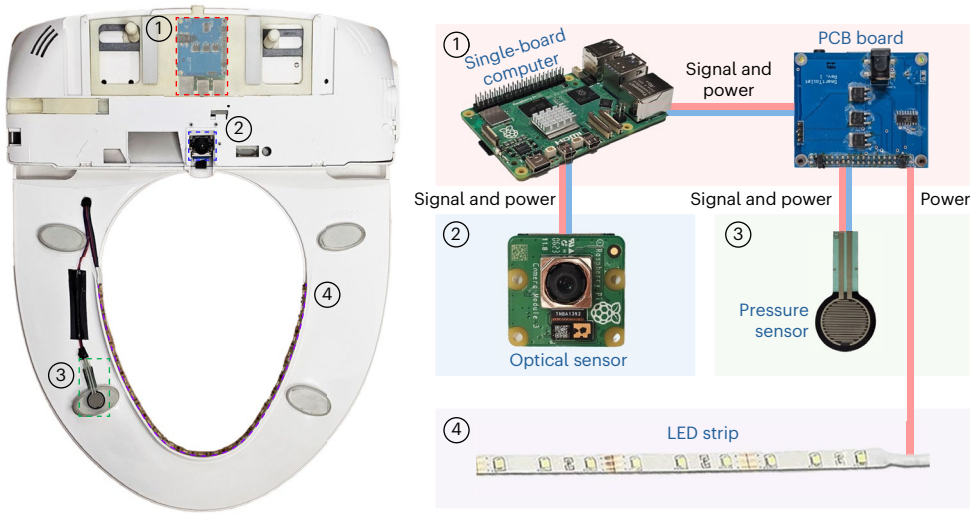

Fig. 3 | Hardware configuration of the PHIND system. The overview of the layout of the constructed toilet seat (left, shown horizontally flipped) and the hardware components within the setup (right). The single-board computer and the PCB (1) give power and receive signals from the optical sensor (2) located at the center of the device housing, with the pressure sensor (3) under the toilet seat and the LED strip (4) on the inner rim of the toilet seat.

outcomes[4], the system enhances the reliability of data used for diagnosing and managing gastrointestinal conditions. Operating passively and noninvasively, the PHIND system collects valuable health information seamlessly in everyday life, promoting long-term user compliance.

Limitations

Although the concept of a smart toilet has been discussed in the medical engineering community for years, adoption has been limited owing to societal taboos surrounding excretion and concerns about privacy[28–30]. User acceptance is critical for the widespread implementation of such a system. Addressing privacy concerns is essential; thus, our team has prioritized secure data handling and user confidentiality in the system's design.

In addition to barriers to user acceptance, one may want to consider the following technological limitations for the system's research use. A key technical challenge is ensuring that all components within the toilet's bidet casing are protected from moisture, as the system has not yet undergone a waterproofing process. This waterproofing feature is planned for future steps toward commercialization. The hardware is also vulnerable to disruptions from strong impacts or shocks when users sit down heavily, potentially affecting internal wiring and connections. Designing robust enclosures and secure mounting systems that absorb impacts could mitigate this issue. In addition, during extreme conditions such as intense defecation episodes (for example, diarrhea), the optical sensor's lens may become soiled or contaminated, impairing stool recognition. Integrating a contamination detection mechanism along with automated cleaning or washing systems for the lens can enhance the system's reliability in real-world applications.

Experimental design

PHIND system

Hardware design and system fabrication: as shown in Fig. 3, the primary objective of the first stage is to develop a hardware system for passive data collection. The process begins with constructing a toilet-mounted system with various sensors designed to identify the user and monitor their activity. A pressure sensor is integrated using jumper wires and an analog-to-digital converter (ADC), MCP3008, to convert the analog pressure signals into a digital form, enabling the detection of the user's presence based on the pressure on the toilet seat. When the detected pressure exceeds a predefined threshold, a light-emitting diode (LED) strip inside the

toilet bowl is triggered to light up, providing consistent, ambient lighting for the optical sensor to capture the defecation activities. The various hardware and sensors of this PHIND system are first tested using a single-board computer and a breadboard.

Once testing is completed successfully on the breadboard, the circuit design is transferred to a printed circuit board (PCB) layout, which is then fabricated by an external manufacturer. There is a small chance that the first version of the PCB may contain minor errors, requiring the user to communicate with the manufacturer for design adjustments. Therefore, we encourage users to first assemble the circuit on a breadboard to become familiar with the overall design and functionality. This step is also beneficial for users who wish to add additional sensors or customize system functions. However, if this step proves too difficult, users may also choose to skip the breadboard testing (Steps 10–36) and directly use the circuit diagrams provided in this paper to commission PCB fabrication from a manufacturer.

System installation and maintenance: the fabricated PCB is then mounted onto a single-board computer to ensure the proper functioning of the pressure sensor, optical sensor, LED strip, mouse, keyboard and fingerprint scanner. This hardware system is integrated into a salvaged commercially available electronic bidet, utilizing the existing bidet cavity, which provides sufficient space for hardware installation. The optical sensor is positioned at the center of the bidet housing, fixed at an angle of ~30° from the horizontal to the toilet water to prevent the optical sensor from capturing any physical features of the user. The pressure sensor is installed under the toilet seat mount, and the LED strip is secured around the inner rim of the toilet bowl to provide consistent lighting and effectively monitor and capture user's activity when seated. Last, the fingerprint scanner is installed in place of the toilet's flush handle to enable passive user identification during flushing when the event is completed.

The hardware components of the PHIND system are fundamentally adaptable to toilets of various shapes and dimensions, as the design facilitates installation directly onto the toilet seat rather than onto the toilet fixture itself. Given that most commercially available toilet seats follow standardized mounting formats, users have the flexibility to select a compatible seat that matches their existing toilet structure. In most cases, identifying whether the toilet is round front (~40–44 cm, measured from the mounting holes to the front edge) or elongated (~46–48 cm, measured from the mounting holes to the front edge) is sufficient to ensure proper compatibility between the toilet and the seat. Considering the high cost and complexity associated with replacing entire toilet fixtures, it is advisable that users measure the dimensions of their current toilet carefully and select a suitable toilet seat accordingly, onto which the PHIND system components can then be integrated.

For toilet cleaning, although the toilet seat casing provides basic water protection for the PHIND system, it is recommended to lift the toilet seat and take care to avoid splashing water onto the LED strip, pressure sensor and optical sensor module. If this precaution is taken, there is no need to remove the entire PHIND system during routine toilet cleaning.

Software and model training: this protocol uses an Internet-of-Things-based system with software designed to provide objective analysis of stool forms. Data collection begins with the acquisition of stool images, which are curated by medical professionals. Using professionally annotated data samples (that is, stool images), these data are calibrated using machine learning techniques and are used to train a CNN model that accurately classifies the stool images.

The software system is structured in a sequence of data processing stages, initiated when the pressure sensor detects user seating and activates the system (Fig. 4a). When activated, the LED strip turns on, and then, the stool images captured by the optical sensor are transferred to the cloud. The process concludes when the pressure sensor is deactivated, at which the system prompts the user for fingerprint identification to label the data accurately. After user identification, the system returns to a standby state until the next event is triggered. Once the data are transmitted to the cloud (AWS S3), the AWS Lambda function automates the real-time analysis of images for a smart health-care system, focusing on stool image classification for gut health monitoring (Fig. 4b). Upon receiving an event from the S3 bucket, the Lambda function processes the image by downloading it to an EC2 instance via AWS Systems Manager (SSM). The function triggers machine learning models (four-class, three-class and seven-class classifiers) hosted on the EC2 instance to analyze the images. The function uses a retry mechanism with

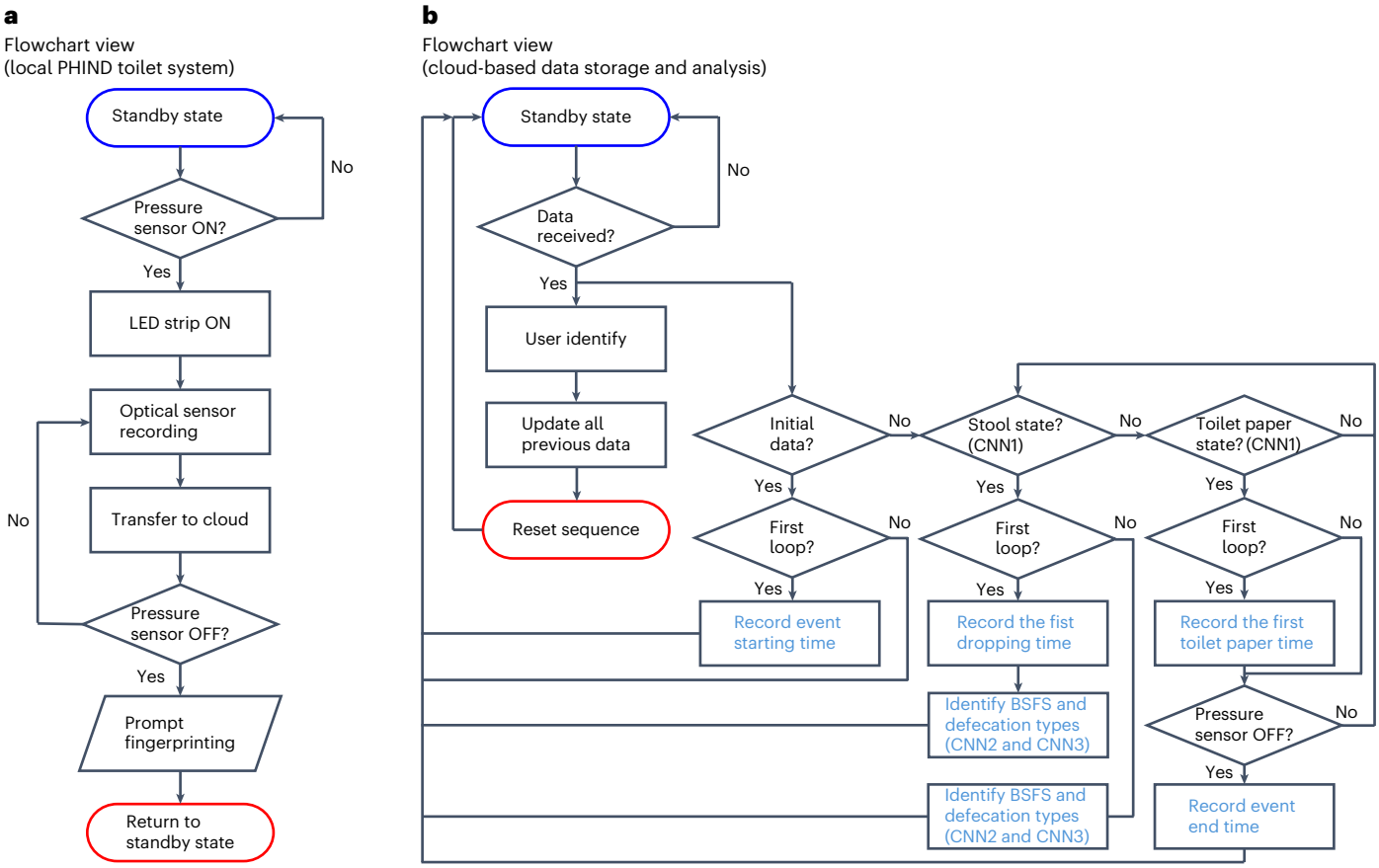

Fig. 4 | Flowcharts showing the use of the PHIND system. a, The process initiates in a standby state and waits for activation from the pressure sensor. Once activated, the optical sensor records stool images, which are then transferred to the cloud for analysis. After the pressure sensor is deactivated, the system prompts the user for fingerprint identification to label data accordingly. The process concludes with the system returning to standby mode until the next event is initiated. **b**, The flowchart shows the sequence of data processing and analysis in the cloud after receiving signals from the local PHIND system in **a**. Once data are received, the user identification is verified and the previous data are updated. If the data are initial, the system records the start of the event. The system then identifies stool state using CNN1 and CNN2 models. Once the pressure sensor detects the event is complete, the system records the event end time and stores the results for data compilation.

exponential backoff to ensure reliable execution. The first set of CNN models (CNN1) utilizes the EfficientNetB0 architecture, designed for a four-class classification system. This system classifies images into four distinct categories: 'clean toilet bowl', 'urine', 'stool' and 'toilet paper' state, representing typical conditions found in toilet settings. Once the optical sensor captures an image, and the model classifies it as the 'stool' state, the system proceeds to a subsequent three-class classification. This second model (CNN2) is also constructed using the EfficientNetB0 architecture, which classifies the stool data into class 1 (constipation), class 2 (normal) and class 3 (diarrhea), providing intuitive information on gastrointestinal condition to the users. Last, the same image data set and architecture are utilized to train and build a seven-class classification model. This last model (CNN3) classifies the stool according to the seven different states outlined in the BSFS.

Once the analysis is complete, the predicted classes and associated probabilities are parsed and stored in an Amazon DynamoDB table for future access. Moreover, JavaScript Object Notation (JSON) files containing metadata, such as user information and event timestamps, are also saved directly to DynamoDB.

Machine learning framework: TensorFlow is an open-source framework developed by Google for building, training and deploying machine learning and deep learning models. Along with PyTorch, it is one of the most widely used platforms in the field of machine learning. In this

work, PyTorch is preferred owing to its superior compatibility with Compute Unified Device Architecture (CUDA) for graphics processing unit (GPU)-based machine learning tasks.

Cost predictions: based on AWS pricing (ap-southeast-1, Singapore region), the operational cost of PHIND system, for a full cumulative hour of defecation events involving multiple users, is ~US$0.220 per hour. This hourly cost encompasses all critical services: an EC2 instance ('c6i.xlarge') running continuously (at US$0.196 per hour), AWS S3 storage for image uploads including the associated PUT (upload operations to store objects in S3) requests (US$0.018 per hour), the S3 standard storage cost for 3,600 images (US$0.000009 per hour), AWS Lambda function triggers to facilitate event-driven interactions (less than US$0.001 per hour) and DynamoDB with provisioned read and write capacity units to handle database operations (US$0.004 per hour). The data transfer costs between S3 and EC2 within the same AWS region are negligible, as AWS does not charge for intraregion communication. The cost model presented here assumes the real-time processing of 3,600 images (~70 kB each, resolution of 640 pixels × 480 pixels), uploaded to PHIND at a rate of one image per second and the proportional resource consumption required by AWS Lambda and DynamoDB under these conditions. This approach demonstrates a highly economical and scalable solution suitable for intermittent workloads. It is also recommended to regularly monitor AWS usage and costs via the Cost and Usage Dashboard, which is accessible directly from the AWS Management Console homepage, to avoid unexpected charges.

Legal and ethical compliance notice

The toilet is universally regarded as one of the most private places, where individuals have an expectation of total privacy. Therefore, any installation of optical sensors or similar devices in such settings must be handled with the utmost care and strictly adhere to all privacy laws and ethical guidelines. This protocol outlines the use of optical sensors in a toilet system for health monitoring by capturing and analyzing defecatory events. It is essential that the installation and use of this system comply with relevant ethical oversight committees, such as institutional review boards (IRBs) or equivalent bodies, to ensure that all privacy concerns are fully addressed.

In institutional or large-scale settings, such as hospitals or care homes, the installation of such systems typically requires formal ethical oversight provided by the IRB or equivalent ethics committees affiliated with the respective institutions. The approval processes generally involve the submission of a detailed protocol to the institution's IRB, clearly describing the procedures for data collection, storage, processing and protection. Obtaining informed consent from each individual user is expected whenever feasible, with clear explanations provided about the nature of the system, the purposes for which data will be used and the measures taken to ensure privacy. Strict data management protocols must be implemented to ensure anonymity or pseudonymization of data, thereby protecting user identities. In addition, transparent communication regarding the system's purpose and potential risks is required. In scenarios where obtaining explicit consent from every individual user is not practically achievable (such as for users with severe cognitive impairment), consent should typically be obtained from their legal guardians or designated proxies.

For individual users installing the system within their private homes, formal IRB approval generally is not necessary because individual users implicitly consent ethically by choosing to install and use the system themselves. Nevertheless, the primary user bears the responsibility for ensuring that other household members or cohabitants are adequately informed about the presence, functionality and data collection practices of the system. Explicit consent from other household members should be obtained, particularly when their data might be captured by the system. Visitors should also be informed by the primary user about the presence and function of the optical monitoring system before its use, and ideally, alternative arrangements should be provided if visitors decline consent.

In both large-scale institutional and individual home scenarios, transparency, clear communication and robust data protection measures are essential components of ethical system installation and operation, ensuring that privacy concerns are comprehensively addressed.

Even when following this protocol, installing the device without the explicit informed consent of all users may result in severe legal consequences. Laws in many countries, such as Article 14 of the Republic of Korea's 'Special Act on the Punishment of Sexual Violence Crimes', Section 377BB of Singapore's Penal Code, the US Video Voyeurism Prevention Act (18 U.S.C. § 1801) and Article 42 of People's Republic of China's 'Public Security Administration Punishment Law', as well as Article 253 of People's Republic of China's 'Criminal Law', prohibit the unauthorized recording or photographing of individuals in private settings, including toilets.

Any modifications or alterations to the system that enable capturing images beyond the intended defecatory events could lead to significant legal liabilities, including criminal charges for voyeurism or privacy invasion. It is important to note that even installing the device as intended, without proper consent and ethical approval, could result in legal consequences. The responsibility for ensuring compliance with these legal and ethical standards rests entirely with the individuals or institutions implementing this protocol. The authors of this protocol do not assume any liability for misuse, unauthorized alterations or noncompliant installations. Implementers are urged to obtain all necessary ethical approvals and ensure that the system is used solely for its intended purpose of health monitoring. For a comprehensive overview of the necessary ethical, legal compliance and data security measures, please refer to the previously detailed guidelines outlined in the manuscript ('Development of the protocol' and 'Legal and ethical compliance notice' sections). Specifically, implementers should follow the instructions provided regarding encryption methods, role-based access control and informed consent procedures, as previously described.

Materials

Equipment

Software

- Raspbian operating system (OS; Raspberry Pi OS (64-bit)) https://www.raspberrypi.com/software/
- PyCharm (PyCharm 2025.1.1.1)—https://www.jetbrains.com/pycharm/#
- AWS S3
  ▲ **CRITICAL** Be mindful of storage costs and data transfer fees, especially with large amounts of data with frequent access.
  ▲ **CRITICAL** This protocol utilizes AWS services and features based on the version updates as of 21 April 2025.
- Lambda (Python)
  ▲ **CAUTION** Monitor execution times and invocations to avoid cost spikes, especially with high-traffic functions.
- AWS EC2 (c6i.xlarge)
  ▲ **CAUTION** Be mindful of the potential cost spikes and ensure to turn off instances after usage.
- Django (4.2.16)

Hardware

- Monitor
  ▲ **CRITICAL** It is recommended to use a monitor with a high-definition multimedia interface (HDMI) port.
- Desktop or laptop
- Toilet (American Standard, cat. no. CL26305-6DACT)
  ▲ **CRITICAL** Ensure the toilet is positioned on a flat, level surface to prevent rocking or breaking the ceramic. Note that the flush button is typically located on the front, side, or top of the water tank. The fingerprint scanner should be positioned accordingly on the basis of the location of the flush button (Step 45).

Protocol

- Single-board computer (Raspberry Pi 5, 8 GB RAM)
  ▲ **CRITICAL** Raspberry Pi 5 is available in two versions: one with 4 GB RAM and another with 8 GB RAM. For optimal performance, the 8 GB version is recommended.
- Optical sensor (Raspberry Pi Camera Module 3)
  ▲ **CRITICAL** The camera module is not waterproof; ensure it is kept in a dry environment to prevent short circuits. The camera module is sensitive to static electricity and physical damage. Always handle with care.
- Red, green, blue (RGB) LED strip with three pins for RGB and one pin for 12V direct current (DC) (Minger Strip Lights, 16.4ft, RGB Color)
- Breadboard for plugging the components (Solderless 830 Tie Point)
  ▲ **CRITICAL** An incorrect placement of components can lead to short circuits and overloading of current that can damage the interior.
- Jumper wires (male to female) for the connection to the Raspberry Pi (jumper wire male–female, 30 cm × 40 cm) (Bus Board Prototype Systems, cat. no. ZW-MF-30)
- Jumper wires (male to male) for connecting the components (jumper wire male–male, 30 cm × 40 cm) (Bus Board Prototype Systems, cat. no. ZW-MM-30)
- Three N-channel metal–oxide–semiconductor field-effect transistor (MOSFETs) for controlling the LEDs (Infineon Technologies, cat. no. IRFZ44N MOSFET 49A 55V)
  ▲ **CRITICAL** Use a gate threshold voltage of maximum 3.3 V (usually marked by an 'L' for Logic-Level in the name).
- Suitable power supply for the LED strip (12 V DC ~2A) (Amazon Wall Adapter Power Supply)
  ▲ **CRITICAL** The power jack must fit with the power supply.
- Pressure sensor (FSREK FSR Model 402)
  ▲ **CRITICAL** As the sensor is a thin-film sensor, too much mechanical stress should be avoided to ensure accurate and consistent readings.
- Optical sensor serial interface cable (Raspberry Pi 5 Camera serial interface flexible flat cable/flexible printed circuit cable cable)
- Mass storage device (universal serial bus (USB) flash drive or hard disk) (SanDisk Ultra 32 GB MicroSD card)
- Wireless keyboard and mouse (Logitech MK235 Wireless combo)
- Screwdriver (Wera 05134000001 Kraftform Micro Big Pack 1 Screwdriver Set, Electronic Applications, 25 pcs)
- Suitable solid support for single-board computer and optical sensor
  ▲ **CRITICAL** Single-board computers such as the Raspberry Pi 5 and optical sensors such as the Raspberry Pi Camera Module 3 come with mounting holes. Users need to design a solid support structure based on the position of these holes and the specific conditions of the toilet casing and then fabricate it using three-dimensional (3D) printing.
- Resistor (1,000 ohm) (BOJACK 1,000 Pcs 25 Values Resistor Kit)
- Fingerprint scanner (optical fingerprint sensor AS608)
  ▲ **CRITICAL** Dirt, oil or moisture on the scanner can affect the accuracy of fingerprint recognition.
- Transistor–transistor logic (TTL) serial to USB converter (CP2102 TTL 5-pin converter)
- ADC (Microchip Technology MCP3008-I/P)
- Secure digital (SD) card adapter (USB 2.0 MicroSD Card Reader)
- Mini HDMI port

Procedure

Stage 1: assembling and mounting the hardware

OS (Raspberry Pi OS) installation

1. Turn on a separate computer (for example, PC or Mac) for the initial installation of the OS.
2. Insert the SD card into the SD card adapter and insert it into the computer (Fig. 5a).
   ◆ **TROUBLESHOOTING**

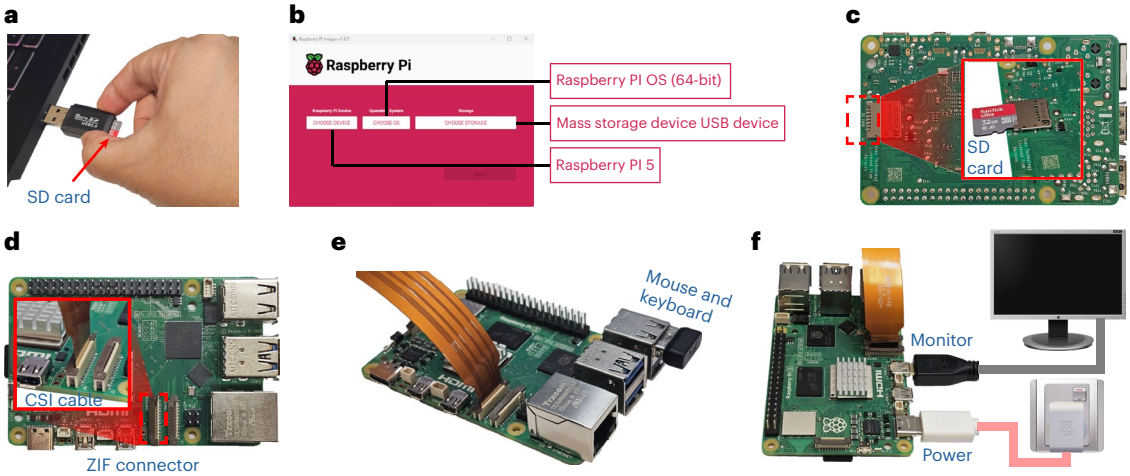

Fig. 5 | Setup of the OS and hardware. a, The installation of the OS (Raspberry Pi5) in the SD card. **b**, The selection of the OS models from the Raspberry Pi Imager. **c**, The insertion of the OS-mounted SD card into the single-board computer. **d**, The camera serial interface (CSI) optical sensor cable connected to the single-board computer at the ZIF connector port. **e**, The connection of the wireless/wired mouse and keyboard using the USB 2.0 port on the single-board computer. **f**, The connection of the hardware-mounted single-board computer to the monitor and 5-V power source.

3. Download the Raspberry Pi Imager from https://www.raspberrypi.com/software/ onto the computer used in Steps 1 and 2.
4. Run the Raspberry Pi Imager to install the Raspberry Pi OS onto the SD (Fig. 5b).
   ▲ **CRITICAL STEP** Select the correct machine model and make sure the Raspberry Pi OS is installed on the SD card.
5. Insert the SD card into the single-board computer (Raspberry Pi 5, Fig. 5c).

Connecting single-board computer with hardware

6. Connect optical sensor module (Raspberry Pi Camera Module 3) to the single-board computer (Raspberry Pi 5) using an optical sensor serial interface cable (Raspberry Pi 5 Camera serial interface flexible flat cable/flexible printed circuit cable; Fig. 5d).
   ▲ **CRITICAL STEP** Open the zero insertion force (ZIF) connector before connecting the cable. The cable has 22 pins on one side and 15 pins on the other. The 22-pin end connects to the Raspberry Pi 5, and the 15-pin end connects to the camera module. Make sure the orientation of the electrical contacts is correct.
   ◆ **TROUBLESHOOTING**
7. Connect the wireless/wired mouse and keyboard to the single-board computer (Raspberry Pi 5) via USB 2.0 ports (Fig. 5e).
8. Connect the monitor via the mini HDMI port (Fig. 5f).
   ◆ **TROUBLESHOOTING**
9. Connect the 5 V power to the single-board computer (Raspberry Pi 5; Fig. 5f).

Circuit test using breadboard (optional: this section can be skipped if one decides to directly fabricate a customized PCB, in which case, proceed to Step 37)

10. Position the breadboard horizontally to point the red power rail toward the bottom and the blue ground to face the top.
11. Identify the rows by capital alphabet letters and the columns by numerical values located between the power rails.
    ▲ **CRITICAL STEP** The central circuits (labeled alphabetically) are connected and flow vertically, whereas the power rails flow horizontally.
12. Install the MCP3008 Converter on the left side of the breadboard, as shown by the number 8 in Fig. 6a. The bottom eight headers are in row E, and the top eight headers are in row F, spanning columns 5–12.

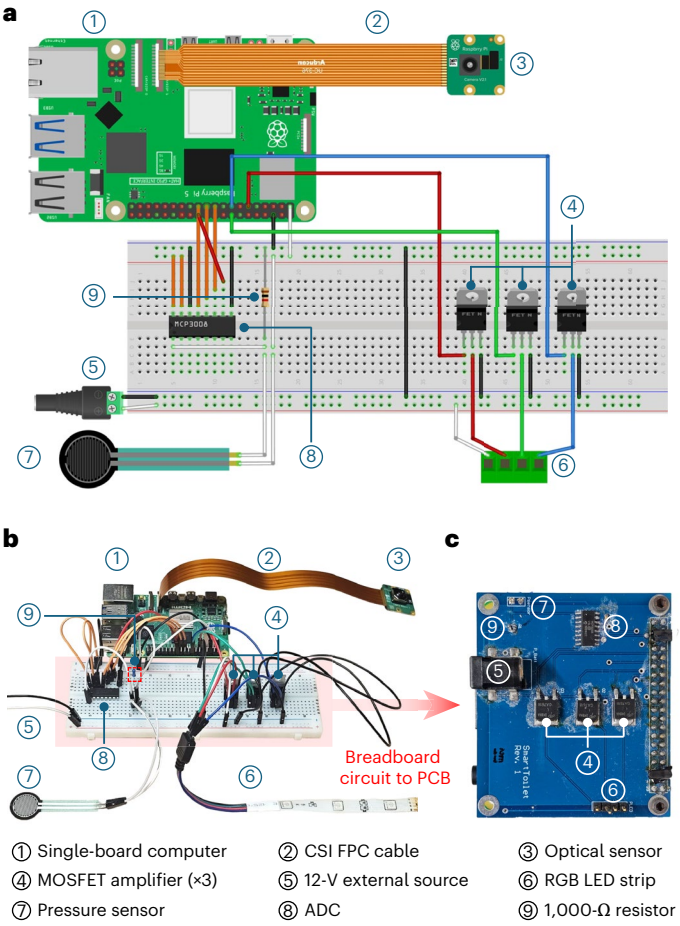

Fig. 6 | Overview of the breadboard and PCB layout. a, An illustrated diagram of the constructed breadboard layout and the hardware. **b**, The actual breadboard constructed with the sensors, resistors, single-board computer and power source connected. **c**, The actual image of the fabricated PCB based on the breadboard constructed.

▲ **CRITICAL STEP** Ensure the correct orientation of the MCP3008 Converter with the small circular groove at the bottom left.

13. Connect one end of the resistor to the top ground (blue) rail and the other end to row E, column 16 (Fig. 6a, number 9).
14. Connect a male–male jumper wire from the top positive (red) rail to row D, column 17.
15. Connect a male–male jumper wire from channel 01 of the MCP3008 to Row D, column 16.
16. Position the pressure sensor across row C, columns 16 and 17 (Fig. 6a, number 7).
17. Connect pins 15 and 16 to the top power rail at columns 5 and 6, respectively.
    ▲ **CRITICAL STEP** For comprehensive information about the MCP3008's channel input for each header, visit https://forge.codesys.com/drv/mcp3008/home/Home/.
18. Attach pin 14 to the top ground rail, at column 7.
19. Connect all pins 13, 12 and 11 to Raspberry Pi general-purpose input/output (GPIO) 11, 9 and10 (serial clock, master in slave out and master out slave in), respectively.
20. Attach pin 10 to Raspberry Pi GPIO 8 (CE0).
21. Link pin 9 with the top ground rail at column 12.
22. Orient all three MOSFET transistors with the silver flat face (heat sink) facing upward and the black curved body (belly) facing row A at the bottom. Ensure that the half-circle groove on each MOSFET is positioned to the left (Fig. 6a, number 4) for proper alignment.
23. Place transistors at row D, columns 40, 46 and 52 with respect to gate pin 1G (left).
24. Designate the left, center and right transistors as red, green and blue, respectively (Fig. 6a).
25. Connect each transistor's center pin (drain) to the corresponding color source pins on the LED strip.

26. Connect the LED strip's power source pin to the bottom power (red) rail.
27. Connect each transistor's rightmost pin (source) to the corresponding column's bottom ground (red) rail.
28. Link each transistor's gate pin (RGB) to Raspberry Pi GPIO 17, 23 and 22, respectively.
29. Connect the 12-V external power source's positive input to the bottom power rail and the negative input to the bottom ground rail.
30. Connect the Raspberry Pi 5's 3V3 power pin to the top power rail.
31. Connect Raspberry Pi 5 ground (pin 6) to the top ground rail.
32. Lastly, use a male–male jumper wire to connect both sides (top and bottom) of the ground rails, establishing a common ground.

Testing sensors and connections (optional: this section can be skipped if one decides to directly fabricate a customized PCB, in which case, proceed to Step 37)

33. Test the camera module by opening the terminal window, entering the command 'sudo raspi-config' to configure the camera, and then entering the 'libcamera-hello' command to test the connection. If the camera is connected successfully, the camera preview window will temporarily appear (see Step 6 for connecting the camera to the single-board computer.)
34. To test the ADC, MCP3008, open a terminal window and enter 'nano' to write a Python code that checks each channel and its corresponding value. If the connection is established correctly, the terminal window will display the channel numbers and their values. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).
    ◆ **TROUBLESHOOTING**
35. To test the pressure sensor, open a terminal window and enter 'nano' to write a Python code that records the pressure value. Then press the pressure sensor. If the connection is established correctly, the terminal window will display the pressure values. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).
    ▲ **CRITICAL STEP** The ADC MCP3008 typically converts the analog signal from the pressure sensor into a range of numbers from 0 to 1023. However, owing to the difference between the reference voltage of the MCP3008 and the maximum output voltage of the pressure sensor, the maximum values recorded by different pressure sensors may vary.
    ◆ **TROUBLESHOOTING**
36. To test the LED strip, open a terminal window and enter 'nano' to write a Python code that lights the red, green and blue color of the LED. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).
    ▲ **CAUTION** There is a risk of electric shock in case of faulty wiring. Do not connect the power source before testing the LED strip.
    ◆ **TROUBLESHOOTING**

Fabrication of PCB

37. Fabricate a PCB based on the breadboard design and specifications explained above. Figure 6a can be provided to the PCB manufacturer for PCB fabrication. (Fig. 6b,c).

Assembling PCB to the single-board computer

38. Connect the PCB from the previous section to the single-board computer used in Steps 5–9 via GPIO headers. This connection establishes an interface between the computer's processing units and the external components mounted on the PCB (Fig. 7a).
39. Mount the LED strip onto the four-pin RGB connector of the PCB.
    ▲ **CRITICAL STEP** Each of the four pins correspond to the wires of the LED strip to ensure proper functionality (Fig. 7b).
40. Plug in the 12-V external LED power source to the DC barrel jack connector on the PCB and ensure stable and secure connection to supply consistent power to the LED strip during the operation (Fig. 7b).

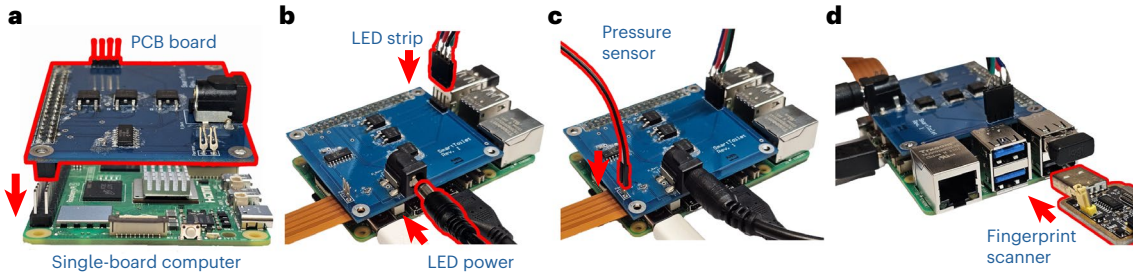

Fig. 7 | Assembling the PCB board to the single-board computer. a, Mounting the fabricated PCB board onto the single-board computer via the GPIO headers. **b**, Connecting the LED strip to the four-pin connector and the LED power source to the DC barrel jack of the PCB. **c**, Connecting the pressure sensor to the two-pin connector using jumper wires. **d**, Inserting the TTL converter of the fingerprint scanner into the USB 2.0 port of the single-board computer.

41. Cut off one end of the female–female or male–female jumper wires, leaving the female end intact, and solder the cut ends to the pressure sensor using a lead solder.
    ▲ **CRITICAL STEP** Ensure to prevent short circuits between the two wires.
42. Mount the pressure sensor onto the two-pin connector using jumper wires (Fig. 7c).
43. Sequentially connect the fingerprint scanner's transmit data (TXD), receive data (RXD), ground and 3V3/5V to the TTL serial to USB converter using jumper wires.
    ▲ **CRITICAL STEP** Ensure that the TXD pin of the fingerprint scanner is connected to the RXD pin of the USB converter and that the RXD pin of the fingerprint scanner is connected to the TXD pin of the USB converter.
    ◆ **TROUBLESHOOTING**
44. Plug in fingerprint scanner into the USB port of the single-board computer (Fig. 7d).
45. Create a case for the fingerprint scanner using a 3D printer, designed to attach the scanner to the toilet's flush button. This will allow the user's fingerprint to be recorded when they flush the toilet after the event. Alternatively, download the 3D model file from GitHub (https://github.com/szq1223/PHIND-system.git).
    ▲ **CAUTION** Note that the flush button is typically located on the front, side or top of the water tank. The shape of the casing should be designed to match the button's location and shape for replacement. The fingerprint scanner should be positioned accordingly on the basis of the flush button's placement.

Assembling the PHIND system to the toilet

46. 3D-print a solid support pedestal to hold the mounted single-board computer and the PCB between its walls. Tighten the walls using screws to ensure that both boards are firmly fastened (Fig. 8a). Alternatively, download the 3D model file from GitHub (https://github.com/szq1223/PHIND-system.git).
47. Install the assembled boards and support structure into the cavity of the bidet case using screws (Fig. 8b).
48. 3D-print a solid casing for the optical sensor to provide protection and to facilitate easier assembly. Enclose the optical sensor inside the casing before installation. Alternatively, download the 3D model file from GitHub (https://github.com/szq1223/PHIND-system.git).
49. Remove the original bidet nozzle from the interior and install the enclosed optical sensor in the same cavity in the bidet casing with screws (Fig. 8c).
    ▲ **CRITICAL STEP** Ensure the optical sensor is precisely aligned and fixed at 30° down from the horizontal to focus only on the intended area inside the toilet bowl. Adjust the setup angle, if necessary, to prevent accidental capture of any body parts and verify the field of view when testing.
    ▲ **CRITICAL STEP** Any modifications or unauthorized adjustments to the optical sensor and its angle may lead to serious privacy violations and legal consequences. It is crucial to adhere strictly to the established protocol to ensure compliance with privacy regulations and to avoid any legal consequences. Unauthorized alterations could compromise the confidentiality of sensitive biometric data and result in significant legal and ethical issues.

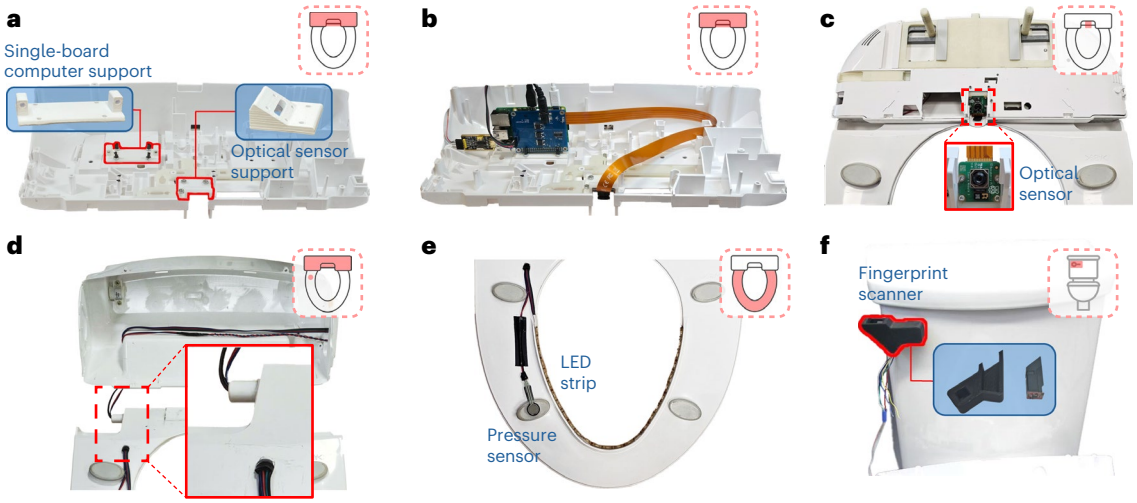

Fig. 8 | Assembling the PHIND system to the toilet. a, Securing the 3D-printed supports for the single-board computer and the optical sensor onto the electronic bidet casing using a screw. **b**, Mounting the assembled single-board computer PCB unit and the optical sensor onto the respective supports. **c**, A close-up view of the optical sensor when the electronic bidet casing is assembled onto the toilet seat. The optical sensor must be ensured to have a fixed angle of 30 degrees to the horizontal. **d**, Arranging the wires for the pressure sensor and the LED strip through a hole to be passed through the bidet casing for the connection to the single-board computer PCB body. The inset shows an enlarged view of the hole through which the wires pass. **e**, Securing the pressure sensor and the LED strip onto the bottom of the toilet seat and around the inner rim, respectively. The exposed electrical wires must be taped and covered with waterproof insulation tape. **f**, The replacement of the original toilet flush with a fingerprint scanner. The fingerprint scanner must be mounted inside the 3D-printed casing to be secured onto the toilet.

▲ **CRITICAL STEP** If the PHIND system is used for academic research and data collection in public restrooms exclusively involving study participants, the system must be removed after all data acquisition and must not be kept installed on the toilet without notice. Any abuse or misuse of the PHIND system and the optical sensor outside notified testing environments will result in significant legal consequences.

50. Arrange the electrical wires (LED, pressure sensor and fingerprint scanner wires) from inside the bidet cavity through the designated opening, ensuring no damage to the cables. This may involve drilling a hole in the plastic cover of the toilet (Fig. 8d).
▲ **CRITICAL STEP** Any drilled holes should be applied with waterproof sealing to ensure the system's durability and safety in a wet environment.
51. Install the pressure sensor under the toilet seat and seal the lose wires with waterproof tape. The sensor should be secured on either side of the front seat bumpers to more effectively capture pressure signals from participants who only sit on the front half of the seat (Fig. 8e).
▲ **CRITICAL STEP** Although the pressure sensor is a thin-film type, there is still a chance that installing the additional sensor may cause an imbalance among the four foot pads of the toilet seat. This could result in an abnormally high pressure reading even when no one is present, making the sensor less sensitive to actual usage. If this issue occurs, adjust the gap between the toilet seat and the bowl by applying tape to the other foot pads to maintain balance. This ensures that the pressure sensor functions properly both before and after participants sit on the toilet.
▲ **CRITICAL STEP** The pressure sensor threshold is empirically set to be 130, but this should be adjusted accordingly with the hardware and the pressure sensor. If threshold is set too high, the LED strip will not be powered on easily.
52. Install the LED strip on the inner rim of the toilet seat. Secure the strip using a glue gun and any loose wires with waterproof tape (Fig. 8e).
▲ **CRITICAL STEP** Any exposure of electric parts and wires should be molded or coated with waterproof sealing, as the PHIND system is exposed to wet environment.
▲ **CRITICAL STEP** Do not spray or clean with water on the system, as the entire system requires waterproofing measures, but this setup is only coated for experimental purposes.

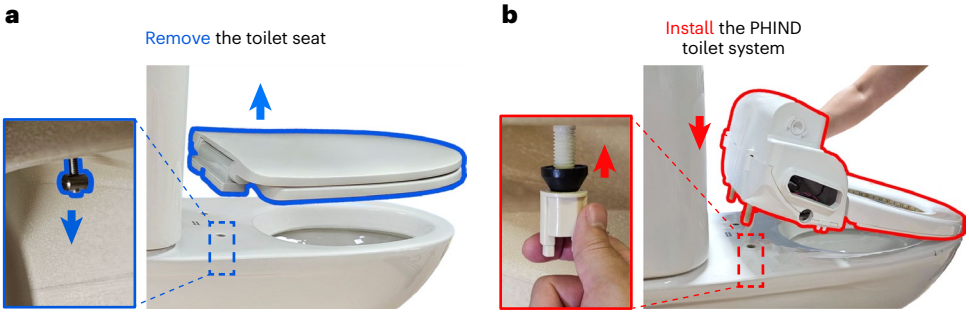

Fig. 9 | Installing the PHIND system onto the toilet bowl. a, Unscrewing the nut from behind the toilet to remove the original toilet seat from the toilet bowl. **b**, Installing the preset bidet seat onto the toilet bowl from the two bolts of the bidet. The seat is secured using screws from behind the toilet.

53. Remove the original flush button of the toilet and replace with a fingerprint scanner in a 3D-printed case, as shown in Fig. 8f (Step 45). Using the eight-pin wire header that is guided out of the bidet cavity, connect the fingerprint scanner with the single-board computer before operation.
54. Connect the alternating current adapter to the regular power outlet (100–240 V depending on the geographic location). Ensure that all the hardware components are securely connected to the PCB-single-board computer body before turning the power on and before running.
    ▲ **CRITICAL STEP** Ensure that the power outlet is adequately rated for the device's voltage requirements (100–240 V) to prevent electrical hazards. Do not use an overloaded or incompatible outlet, as it may result in damage to the equipment.
55. First, remove the original toilet cover and seat for installation of the PHIND system. Begin by unscrewing the nuts from behind the toilet bowl by hand. Once loose, lift the toilet seat cover upwards from the seat (Fig. 9a).
    ▲ **CAUTION** Depending on the model of the toilet, the mechanism for removing the seat cover may vary. Always refer to the manufacturer's manual for specific instructions to avoid damaging the hardware.
56. Align the PHIND bidet unit with the mounting holes on the toilet. From underneath the toilet, use the bidet's bolts to tighten the screws. Ensure that the system is properly aligned and fastened securely before usage (Fig. 9b).

Stage 2: training AI models to identify the condition of the toilet bowl and stool

▲ **CRITICAL** Once trained, the model can be applied across various users and toilet configurations.

57. Prepare a Windows OS desktop or laptop.
    ▲ **CRITICAL STEP** If the computer is using other OS such as Linux, iOS and so on, please change the path in the code provided below. The code assumes that the OS is Windows and that the data are stored on the C drive.
58. Collect images representing four distinct states of the toilet bowl: clean, containing urine only, containing stool (urine may exist) and containing toilet paper. Save them in four separate folders. For each category, it is recommended to collect more than ten images.
    ▲ **CRITICAL STEP** As this step may pose challenges for some users, a set of images has been uploaded to GitHub (https://github.com/szq1223/PHIND-system.git) that can be used for basic model training. This means that any user can train a functional AI model by following this protocol and using the provided sample images. To further improve the model's accuracy, users may collect additional stool images over time. If additional training data are required, please ensure that their collection complies with applicable local laws.
    ▲ **CRITICAL STEP** The image collection does not necessarily have to come from the PHIND system itself. We encourage users to collect the required images from the internet

or capture them using their own cameras. This does not affect the accuracy of the AI model when deployed on the PHIND system and also makes it easier to obtain sufficient training data.

▲ **CRITICAL STEP** To use the code provided below correctly, ensure that the images of the clean, containing urine only, containing stool and containing toilet paper state are saved in the respective folders within the C:\4class directory.

59. Collect stool images from the internet or capture them using own camera and classify them into Bristol scale (BS)1 to BS7 folders on a desktop or laptop by trained clinicians who were trained on specific tasks (that is, BSFS classifications). Save them in seven separate folders. For each category, it is recommended to collect more than ten images.

    ▲ **CRITICAL STEP** As this step may pose challenges for some users, a set of images has been uploaded to GitHub (https://github.com/szq1223/PHIND-system.git) that can be used for basic model training. This means that any user can train a functional AI model by following this protocol and using the provided sample images. To further improve the model's accuracy, users may collect additional stool images over time. If additional training data are required, please ensure that their collection complies with applicable local laws.

    ▲ **CRITICAL STEP** To use the code provided below correctly, ensure that images of stools corresponding to the BS types 1–7 are saved in the BS1 to BS7 folders within the C:\7class directory.

60. Combine the BS1 and BS2 images into a folder named Class1 (constipation), BS3 to BS5 into a folder named Class2 (normal) and BS6 and BS7 into a third folder named Class3 (diarrhea).

    ▲ **CRITICAL STEP** To use the provided code below correctly, ensure that the stool images are saved into the Class1 to Class3 folders within the C:\3class directory.

61. Install a Python integrated development environment (IDE, PyCharm 2025.1.1.1) on a desktop or laptop.
62. Open PyCharm and create a Python project.
63. Right-click on the '.venv' folder under the project and create a new Python file.
64. Open the terminal window of the Python file and enter the following commands to download the necessary libraries.

```
» pip install torch torchvision torchaudio
» pip install opencv-python
» pip install pillow
» pip install numpy
» pip install scikit-learn
» pip install matplotlib
» pip install seaborn
» pip install tqdm
```

    ▲ **CAUTION** Replace the command 'pip install torch torchvision torchaudio' with 'pip install tensorflow' for TensorFlow users.

65. Build a CNN named 'four-class model' using EfficientNetB0 to classify the state of the toilet bowl into four categories: clean, containing only urine, containing stool and containing toilet paper. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git). When using the code, please check the training and validation image paths, as well as the model's name and its save path. The EfficientNetB0 has been tested to give the highest accuracy when classifying the toilet bowl condition images. In the four-class classification task, the accuracy can exceed 98% when the total number of training images is ~320 (~80 for each class).

    ▲ **CRITICAL STEP** It is recommended that 80% of the images be used for training and 20% for validation. To prevent data leakage during training, separate the training images and validation images into different folders, or use code such as: train_samples, val_samples, train_labels, val_labels = train_test_split(samples, labels, test_size=0.2, stratify=labels, random_state=42)

    ◆ **TROUBLESHOOTING**

66. Similar to Step 65, train a 'seven-class model' using EfficientNetB0 to classify the BSFS into seven categories: BS1 to BS7. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git). In the seven-class classification task, the accuracy can exceed 86% when the total number of training images is ~2,300.
◆ **TROUBLESHOOTING**
67. Similar to Step 65, train a 'three-class model' using EfficientNetB0 to classify the defecation condition into three categories: class 1, class 2 and class 3, which represent constipation, normal and diarrhea, respectively. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git). In the three-class classification task, the accuracy can exceed 94% when the total number of training images is ~2,300.
◆ **TROUBLESHOOTING**

Connecting the hardware to the cloud (AWS)

68. Connect the single-board computer to the internet via the Wi-Fi module (Raspberry Pi 5 supports onboard Wi-Fi).
▲ **CRITICAL STEP** Please check the Wi-Fi strength in the experimental environment before the setup.
69. Open the terminal window and run the following commands to upgrade the system.

```
» sudo apt update
» sudo apt upgrade
```

70. Run the following command to install python.

```
» sudo apt install python3
```

71. Create a virtual environment for the PHIND system.

```
» python3 -m venv /home/user_name/PHIND_env
```

72. Active the PHIND_env virtual environment.

```
» source /home/user_name/PHIND_env/bin/activate
```

73. Install the necessary libraries.

```
» pip install lgpio
» pip install pyfingerprint
» pip install matplotlib
```

74. Write a code to activate the entire system when a pressure sensor reaches a threshold value. The appropriate threshold value is ~30 % of the maximum value. Alternatively, download it from GitHub (https://github.com/szq1223/PHIND-system.git).
75. Write a code that turns on the LED strip when the pressure sensor reaches a threshold value set in Step 74. Alternatively, download it from GitHub (https://github.com/szq1223/PHIND-system.git).
76. Write a code to enroll a user's fingerprint that will identify the user. Alternatively, download it from GitHub (https://github.com/szq1223/PHIND-system.git).
77. Write a code that continuously captures images (a 1-s interval is recommended) using an optical sensor when the pressure sensor reaches the threshold value set in Step 74 and uploads the captured images and the associated metadata to an AWS S3 bucket for storage once the event is complete (defined either by scanning a fingerprint or when the pressure sensor returns to a null value for 30 s). Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).

▲ **CRITICAL STEP** The code provided in GitHub requires S3 bucket information. After setting up the S3 bucket (Step 79), fill in the necessary details (Step 81). To alleviate the computational load on the EC2 instance, the current implementation sets the image capture interval to 1 s, with image filenames formatted as YYMMDDHHMMSS (year, month, day, hour, minute and second). Although this interval is adequate for standard defecation analysis, achieving higher temporal resolution would necessitate adjustments, including modifying the image-naming convention, increasing the capture frequency and migrating from libcamera-still to the picamera2 library. However, without a solid coding foundation, it is not recommended to modify the code directly, as doing so may introduce critical bugs, given that subsequent modules are tightly coupled with the existing naming convention.

Setting up core AWS services: S3 Bucket, SQS, EC2 instance, DynamoDB and Lambda function

▲ **CRITICAL** Detailed instructions on the AWS set up can be found in Supplementary Figs. 1–6.

78. Create an AWS account, and then, sign in to the management console with the credentials. In the search bar at the top of the console, search and select 'S3' under 'Services'.
79. To create S3 bucket from the S3 dashboard, select 'Create bucket' and under 'General configuration', choose 'General purpose'. Provide a unique bucket name following the AWS S3 bucket-naming rules (Supplementary Fig. 1a).
80. For the ownership, access, version and encryption options below, it is recommended that all settings are kept to default. Then click the 'Create bucket' (Supplementary Fig. 1b).
    ▲ **CRITICAL STEP** It is recommended that all public access is blocked for security, but this can be amended depending on the user environment (see 'Legal and ethical compliance notice' section of the 'Experimental design' section for further details).
81. Record the access key and secret access key of the S3 bucket. Then, open the terminal window of the single-board computer to write the commands below, connecting the single-board computer to the S3 bucket.

```
» sudo apt-get install awscli
» pip3 install boto3
» aws configure
» AWS Access Key ID: <Enter the Access Key ID you obtained from the AWS
Management Console.>
» AWS Secret Access Key: <Enter the secret access key you received.>
» Default region name: <Enter the AWS region you are using, for example,
us-east-1.>
» Default output format: <You can choose json or leave it blank.>
```

82. To efficiently manage message queuing and prevent data loss during high-speed and large amount of data flux, set up Simple Queue Service (SQS) through the SQS dashboard.
83. Click on the 'Create queue' button, select the 'Standard' type and provide with a unique name for SQS (Supplementary Fig. 2a).
    ▲ **CRITICAL STEP** If first-in, first-out type is selected, the sequence strictly abides by the order but may result in performance bottlenecks and unnecessary restrictions to hinder operation.
84. Under 'Configuration', adjust the 'Visibility timeout' to 30 min (Supplementary Fig. 2b).
    ▲ **CRITICAL STEP** This should be kept longer than the Lambda function Timeout.
85. Maintain the rest of the settings to default and click on 'Create queue' to finish the process (Supplementary Fig. 2c).
86. To grant access permissions to SQS queue, select the created SQS among the list of queues under SQS. Click on 'Edit' button to edit the 'Access policy' (Supplementary Fig. 2d).
87. Under the 'Access policy' section, there is a prewritten JSON to define the accessibility of the queue. Here, edit the code as following for sufficient access required to use the queue, then click on 'Save' (Supplementary Fig. 2e):

```
{
"Version": "2012-10-17",
"Id": "__default_policy_ID",
"Statement": [
{
"Sid": "__owner_statement,
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::<Your_AWS_Account_ID>:root"
},
"Action": "SQS:*",
"Resource": "arn:aws:sqs:<Your_AWS_Region>:<Your_AWS_Account_
ID>:<Your_Queue_Name>"
},
{
"Effect": "Allow",
"Principal": {
"Service": "s3.amazonaws.com"
},
"Action": "SQS:SendMessage",
"Resource": "arn:aws:sqs:<Your_AWS_Region>:<Your_AWS_Account_
ID>:<Your_Queue_Name>",
"Condition": {
"ArnEquals": {
"aws:SourceArn": "arn:aws:s3:::<Your_S3_Bucket_Name>"
}
}
}
]
}
```

88. To grant S3 access to SQS for real-time monitoring of data input, navigate to the S3 bucket and click on the S3 bucket created in Step 80. Navigate to the 'Properties' panel and scroll down to the 'Event notification' section to click on 'Create event notification' (Supplementary Fig. 2f).
89. Under 'General configuration', provide a unique event name (Supplementary Fig. 2g).
90. Under 'Event types', select the option 'All object create events' (Supplementary Fig. 2h).
91. Under 'Destination', select the option 'SQS queue', choose the respective SQS queue created in Step 85, then finish the process by clicking on 'Save changes' (Supplementary Fig. 2i).
92. To create the EC2 instance, search and select 'EC2' in the search bar.
93. On the EC2 dashboard, select 'Launch instance' (Supplementary Fig. 3a).
94. Create a unique name for EC2 instance, then navigate to the 'Quick start' panel under 'Application and OS images' section (Supplementary Fig. 3b).
95. Under 'Amazon Machine Image (AMI)', select the option labeled 'Amazon Linux 2023 AMI' (Supplementary Fig. 3b).
96. Under 'Instance type', select 'c6i.xlarge' (Supplementary Fig. 3c).
97. Under 'Key pair (login), click on 'Create new key pair'. Provide a unique name for the key pair, while maintaining the default settings under 'type' and 'format' (Supplementary Fig. 3d).
98. Finish this operation by clicking 'Create key pair'. This will automatically download the .pem key file, which is used for secure shell (SSH) access to the instance (Supplementary Fig. 3d).
99. Back at the 'Launch an instance' window, leave the rest of the settings as default, set the storage to 32 GiB under 'Configure storage', then click on the 'Launch instance' button to finish the EC2 setup process (Supplementary Fig. 3e).

100. To grant access to the EC2 instance, search for and select 'Identity and Access Management (IAM)' in the search bar to create a new role.
▲ **CRITICAL STEP** To reduce cloud computing costs, it is recommended to terminate the AWS EC2 instance during periods of prolonged inactivity, such as overnight sleep time, especially if the system is intended for single-user purposes. As described in the 'Experimental design' section, the EC2 instance (c6i.xlarge) incurs a cost of US$0.196 per hour while running. To stop the instance when not in use: (1) log in to the AWS Console; (2) click 'EC2', then click 'Instances (running)'; and (3) choose your instance, click 'Instance state', then click 'Stop instance'. To restart the instance: (1) click 'EC2' and (2) click 'Instances', choose your desired instance and click 'Start instance'.
101. In the IAM dashboard, navigate to 'Access management', click on 'Roles' and then click on 'Create role' (Supplementary Fig. 4a).
102. Set the 'Trusted entity type' to 'AWS services' and select 'EC2' for 'Service or use case'. Keep the rest of the settings default, then click 'Next' (Supplementary Fig. 4b).
103. Under 'Permission policies', search for and select the following policies and click 'Next' (Supplementary Fig. 4c):

```
[AmazonDynamoDBFullAccess]
[AmazonSSMManagedInstanceCore]
[AmazonSSMFullAccess]
[AmazonS3FullAccess]
```

▲ **CRITICAL STEP** Although full access can ensure the operation of the entire system, it poses risks such as data leakage, tampering and unnecessary resource consumption. Therefore, it is recommended that permissions be granted according to the principle of least privilege. For instance, read only access is preferable to full access when write access is not required. Users who are highly familiar with AWS may modify access permissions when necessary.
104. To create DynamoDB, search and select 'DynamoDB' in the search bar.
105. Under DynamoDB dashboard, select 'Create table' (Supplementary Fig. 5a).
106. Provide a unique name for the table according to the naming rules (Supplementary Fig. 5b).
107. Under 'Partition key' section, also give a unique name for the partition key to retrieve items in later steps (Supplementary Fig. 5b).
▲ **CRITICAL STEP** The 'Partition key' can be given any unique name, but to use the code provided on GitHub directly, the 'Partition key' must be named 'image_name'.
108. Keep the rest of the settings default and finish this operation by clicking 'Create table' below (Supplementary Fig. 5c).
109. To create a Lambda function for detecting any uploads on the S3 bucket, analyzing and sending those to EC2 instance, search and select 'Lambda' in the search bar then click on 'Create function' (Supplementary Fig. 6a).
110. Select the option 'Author from scratch', then provide a unique function name according to the naming rules (Supplementary Fig. 6b).
111. Select 'Runtime' as 'Python 3.12' and 'Architecture' as 'x86_64' (Supplementary Fig. 6c).
▲ **CRITICAL STEP** Ensure to select any recent versions of Python when configuring 'Runtime' to maintain compatibility and access the recent features.
112. Click on 'Create function' below to finish this process (Supplementary Fig. 6d).
113. To grant access permissions to Lambda, select the created Lambda function, navigate to 'Configuration' and then to 'Permissions' (Supplementary Fig. 6e).
114. Click on the automatically generated 'Role name' to add more permissions to the Lambda function. This will redirect to the IAM Roles associated with the Lambda function that was just created (Supplementary Fig. 6e).
115. Under 'Permissions policies', click on the dropdown button labeled 'Add permissions' and select 'Attach policies' (Supplementary Fig. 6f).
116. Under 'Other permission policies', search for and select the following policies (Supplementary Fig. 6g):

```
[AmazonDynamoDBFullAccess]
[AmazonEC2FullAccess]
[AmazonS3FullAccess]
[AmazonSSMManagedInstanceCore]
[AmazonSSMFullAccess]
```

▲ **CRITICAL STEP** Although full access can ensure the operation of the entire system, it poses risks such as data leakage, tampering and unnecessary resource consumption. Therefore, it is recommended that permissions be granted according to the principle of least privilege. For instance, read only access is preferable to full access when write access is not required. Users who are highly familiar with AWS may modify access permissions when necessary.

117. After seeing the message verifying that the policies were successfully attached to the role, navigate back to the Lambda dashboard and select the respective Lambda function.
118. Under 'General configuration', click on 'Edit' and set a 'Timeout' time to define the maximum amount of time that the Lambda function is allowed to run before automatic termination. Change the default Timeout setting to 10 min (Supplementary Fig. 6h).
119. Finalize this process by clicking on 'Save' (Supplementary Fig. 6h).
▲ **CRITICAL STEP** AWS Lambda pricing is based on the number of requests and the duration of code execution. Although it is cost-effective for short, infrequent tasks, high volumes of requests or long-running functions may lead to unexpectedly high charges. Carefully monitor the usage and optimize the function performance to avoid excessive costs.
120. As SQS needs to be watched over by the Lambda function for any inputs, the SQS queue should be attached to the Lambda function as a trigger. Click on 'Add trigger', and from the dropdown menu, search for and select 'SQS' as the source (Supplementary Fig. 6i).
121. Write a Lambda function that handles SQS messages triggered by new image or JSON file uploads to an S3 bucket, runs image classification on an EC2 instance using trained AI models (four-, three- and seven-class models) and saves the results (predicted classes and probabilities) along with the JSON file data to DynamoDB. Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).
▲ **CRITICAL STEP** The code provided in the GitHub requires the name of the inference script (Step 130). This can be left blank initially and filled in after the script is set up in Step 130.

Stage 3: image acquisition and deploying cloud infrastructure for real-time analysis, data storage and visualization

Image acquisition and storage

122. To check image acquisition on the PHIND system, first sit on the toilet seat and ensure that the LED strip turns on. Then, verify that the image is correctly saved in the /home/<your-username>/images folder. Check whether the image filename follows the format YYMMDDHHMMSS (year, month, day, hour, minute, and second) and confirm that it matches the actual capture time. In addition, test both process termination methods: (1) ending the session using the fingerprint scanner, which should generate a JSON file, and (2) automatically ending the session 30 s after the user leaves the seat, which does not generate a JSON file. Confirm that both modes successfully terminate the process.
▲ **CRITICAL STEP** For now, the process runs locally. Its workflow is as follows: when the pressure sensor detects a value above the threshold, the LED strip is turned on first, followed by image capture. If the pressure drops below the threshold, a 30-s countdown is triggered. If pressure above the threshold is detected again within those 30 s, the countdown is cancelled, and the process continues. If a fingerprint is detected within the 30 s, the process is terminated, and the user identity is recorded. If neither sufficient pressure nor a fingerprint signal is received within 30 s, the system automatically stops the process.

123. To check whether the acquired images are successfully uploaded to the S3 bucket, go to the S3 bucket created in Step 80 and verify that all images and JSON files have been uploaded correctly.

Data analysis and result output in the cloud (AWS)

124. Go back to the EC2, select the created EC2 instance and click 'Connect'.
125. Create a virtual environment for the PHIND system with the commands below.

```
» cd /home/ec2-user/# Navigate to the home directory of the ec2-user
» sudo yum install python3#installing Python3
» sudo yum install python3-pip#installing pip
» python3 -m venv phind_env# Create a Python virtual environment named
'phind_env'
```

126. Activate the PHIND virtual environment (phind_env) with the commands below.

```
» source /home/ec2-user/phind_env/bin/activate#activating virtual
environment
```

127. Install the necessary packages with the commands below.

```
» pip3 install torch torchvision torchaudio#Installing PyTorch and
related libraries using pip
» pip3 install boto3#installing boto3
» pip3 install pandas# Install pandas for data manipulation and
analysis
```

128. Create a folder to save and deploy the AI models (four-class, three-class and seven-class models) with the commands below.

```
» deactivate#Deactivate the virtual environment
» cd /home/ec2-user/#back to ec2-user home
» mkdir models#creating the folder named 'models'
```

129. On the local environment, open the Windows command prompt or PowerShell terminal to upload AI models (four-class, three-class and seven-class models) located in the local directory. Using the commands below.

```
» ssh -i /your/key/path/<your_key_name>.pem ec2-user@your_server_ip#
Use SSH to connect to an EC2 instance
» scp -i /your/key/path/<your_key_name>.pem your/model/path/ <your_
model_name>.pth ec2-user@your_server_ip:/home/ec2-user/models/
# Use SCP to transfer files to the EC2 instance
```

▲ **CRITICAL STEP** It is crucial to monitor the usage metrics and service costs closely, as different service configurations and resource consumption may result in sudden and significant increases in charges. Users should set up cost management strategies and have consistent price watch over the course of the set up (see 'Costs predictions' section in 'Experimental design' section for further information).

130. Write an inference script at /home/ec2-user/models/ to analyze the uploaded image using AI models (four-, three- and seven-class models). Alternatively, download the code from GitHub (https://github.com/szq1223/PHIND-system.git).

◆ **TROUBLESHOOTING**

Create a visible UI using web framework (Django)

131. Set up a Django web project on AWS EC2 instance within the PHIND virtual environment (phind_env).

```
» source /home/ec2-user/phind_env/bin/activate
» pip install django matplotlib
» django-admin startproject phindproject
» cd phindproject
» python manage.py startapp analysis
```

132. Create the folder structure as shown below. The settings.py, dynamodb.py, urls.py (main uniform resource locators [URLs]), urls.py (app URLs), views.py and display_results.html code have been uploaded to GitHub (https://github.com/szq1223/PHIND-system.git).

```
phindproject/
├── manage.py          # Manage script
├── static/            # Static files
├── phindproject/
│   ├── __init__.py    # Init file
│   ├── asgi.py        # ASGI config
│   ├── settings.py    # Settings
│   ├── urls.py        # Main URLs
│   └── wsgi.py        # WSGI config
└── analysis/
    ├── __init__.py    # Init file
    ├── admin.py       # Admin config
    ├── apps.py        # App config
    ├── dynamodb.py    # DynamoDB interactions
    ├── migrations/    # Database migrations
    ├── models.py      # Data models
    ├── tests.py       # Tests
    ├── views.py       # Views
    ├── urls.py        # App URLs
    └── templates/
        └── analysis/
            └── display_results.html    # HTML template
```

133. Write the commands below to run the Django server.

```
» source /home/ec2-user/phind_env/bin/activate
» cd /home/ec2-user/phindproject/
» python manage.py runserver 0.0.0.0:8000
```

134. Enter http://your_ec2_public_ip:8000/analysis/results/ to view the analyzed results.
◆ **TROUBLESHOOTING**

Troubleshooting

Troubleshooting advice can be found in Table 1.

Table 1 | Troubleshooting table

Step	Problem	Possible reason	Solution
2	The MicroSD card is not writable	The SD card adapter's write protection switch is in the wrong position	Move the slider on the SD card adapter
6	The camera is not detected	The cable is not properly connected	With the Raspberry Pi turned off and power disconnected, remove the camera ribbon cable, check the orientation of the pins toward the pins in the socket and reconnect
8	The screen remains black	The components are powered in the wrong order	Power the screen first and the Raspberry Pi second
34–36	The single-board computer cannot control the device connected via GPIO	The GPIO (Broadcom (BCM)) number is confused with the board number of the pins in the Python code	Check the pin position and its GPIO (BCM) number in the code and consistently use the GPIO (BCM) number for coding
36	The LED strip does not light up	The power plug for the LED strip is inserted in the wrong direction	Reverse the direction and reinsert it
43	The fingerprint scanner has no signal or will not turn on	Incorrectly connected the TTL serial to USB converter or set the converter's voltage wrong	Make sure the TXD pin of the fingerprint scanner is connected to the RXD pin of the USB converter and the RXD pin of the fingerprint scanner is connected to the TXD pin of the USB converter. In addition, check whether the output voltage of the USB converter is compatible with the fingerprint scanner
65–67	The GPU cannot be used for machine learning	The computer is missing the necessary parallel computing framework or libraries, or they are incompatible	If using an NVIDIA GPU, ensure that CUDA 12.6 and CUDA deep neural network (cuDNN) 9.3 are installed. If it still does not work, consider switching the machine learning framework (for example, TensorFlow) to PyTorch
65–67	The validation accuracy is too low	Either the training data are insufficient, or the model needs fine-tuning	Increase the training dataset or give more weight to the categories with less training data. Alternatively, adjust the learning rate, or freeze more layers
65–67	The training code either will not run or is showing a warning about loading the model	Difference in TorchVision version	In the # Load the best complete model for testing section, add weights_only=False to ensure the full model is properly loaded
130	Inference script does not work	The CNN model was trained on a GPU, but the EC2 instance is only powered by a CPU	Switch the EC2 instance to a GPU-enabled version or modify the inference script to run inference on the CPU
134	Django does not work	The folder structure in the Django project is not set up correctly	Strictly follow Step 132 to create the file

Timing

Steps 1–5, OS (Raspberry Pi OS) installation: 1 h
Steps 6–9, connecting single-board computer with hardware: 0.5 h
Steps 10–32, circuit test using breadboard: 1.5 h
Steps 33–36, testing sensors and connections: 0.5 h
Step 37, fabrication of PCB: 15 d
Steps 38–45, assembling PCB to single-board computer: 2 h
Steps 46–56, assembling the PHIND system to the toilet: 3 h
Steps 57–67, training AI models to identify the condition of the toilet bowl and stool: 2–20 h (depends on computer hardware)
Steps 68–77, connecting the hardware to the cloud (AWS): 1–3 h (depends on operator's experience)
Steps 78–121, setting up core AWS services: S3 bucket, SQS, EC2 instance, DynamoDB and Lambda function: 2–12 h (depends on operator's experience)
Steps 122–123 image acquisition and storage: 0.5 h
Steps 124–130, data analysis and result output in the cloud (AWS): 1–3 h (depends on operator's experience)
Steps 131–134, create a visible UI using web framework (Django): 1–5 h (depends on operator's experience)

Anticipated results

Upon the successful completion of this protocol, the PHIND system functions seamlessly during the user's routine toilet use. When the user sits on the toilet, the pressure sensor detects the event by surpassing the preset threshold, triggering the system's optical sensor and LED lighting for stool image capture. The fingerprint scanner simultaneously authenticates the user when the event ends and the toilet is flushed, ensuring the correct association of data with the individual. The screen of the single-board computer is shown in Fig. 10a, where the pressure sensor values and real-time data acquisition from the PHIND system can be seen. The LED strip activates immediately, but image capture is delayed by 0.5–1 s owing to the limited computational power of the single-board computer (Raspberry Pi 5). Although the system

Fig. 10 | PHIND system UI. a, The screen shows a sample screen of the pressure sensor values and real-time data acquisition from the PHIND system. **b**, The acquired data analyzed within the cloud system. From top to bottom, the figures represent the pressure sensor graph to identify the start and end time of event, the four-class model (CNN1) to classify the toilet bowl condition, the three-class model (CNN2) to classify the defecation condition and the seven-class model (CNN3) to classify the BSFS class. **c**, An emulated screen of the UI after the analyzed data are transferred from the cloud system.

can handle most defecation cases, extreme cases—such as defecation occurring within 1 s of sitting down or even before sitting—may trigger a red flag. These data can be classified as part of the anomaly dataset such as 'urgent'. The continuous image capture was designed with a 1-s interval to balance detection accuracy and system cost. According to 55 defecation tests, we found that the average total defecation duration (from sitting to water flushing) is 147.1 s (s.d.: 51.3 s), meaning that ~150 images are captured per event[31]. Therefore, a 1-s delay or error would introduce less than a 1% deviation, which is negligible and does not affect the analysis of defecation events.

To improve the response speed from user detection to image capture, this can be easily achieved by upgrading the single-board computer or by continuously monitoring the toilet condition even when the toilet is not in use. The accuracy of defecation time recording can also be improved by modifying the image capture code to enable a shorter capture interval. However, both continuous monitoring and increasing the image capture frequency would require upgrading the EC2 instance, which would increase costs. Although we have added an SQS to reduce the computational load on the EC2 instance, the large volume of continuous data processing and increasing the capture frequency beyond 1 fps would require a high-performance or GPU-enabled instance, which would be three to five times more expensive than the instance recommended in this protocol.

Following image acquisition, stool images are automatically transmitted to a cloud-based storage and computing system for further analysis. This entire process—from event detection to data transmission—is passively and autonomously managed by the system, minimizing disruption to the user's normal toilet routine while ensuring accurate and continuous data collection. All functions, including sensor control, image capture and system communication, are coordinated by a single-board computer installed within the bidet case.

With the analyzed data from the three CNN models and the output signal from the pressure sensor, various parameters of a defecation event can be calculated. As shown in Fig. 10b, from top to bottom, the figures represent the pressure sensor graph, the four-class, three-class and seven-class model outputs. The total event time is calculated on the basis of the period during which the pressure sensor value remains above the threshold. The defecation time is determined by the duration of time when the four-class model identifies stool in the toilet bowl. The first stool drop time is recorded from the moment the four-class model detects the transition from a clean toilet to stool. The defecation condition (constipation, normal or diarrhea) and the detailed BSFS classification are provided directly by the three-class and seven-class models, respectively.

Finally, the parameters are integrated and visualized through a web framework. As shown in Fig. 10c, the analyzed parameters are presented to the user via an intuitive mobile interface. Users can access real-time and historical data on their defecation patterns, with daily summaries of stool classifications and longitudinal graphs tracking defecation conditions.

The seamless integration of hardware and software components, along with the successful demonstration of all functionalities in the relevant environment, shows that the current proposed system would reach technological readiness level 6 (TRL 6). There are nine TRL levels in total, with higher levels indicating greater technological maturity. The detailed definitions for each level can be found at NASA's official page: https://esto.nasa.gov/trl/. This indicates that the system has progressed beyond proof-of-concept and component testing up to where the entire system operates effectively in laboratory conditions that closely simulate the real-world environments. Achieving TRL 6 reflects the system's operational capability and guides toward real-world application. With further refinement in privacy protection and hardware development (waterproofing, robustness and self-cleaning), the system can achieve a higher TRL. It will then be ready for broader clinical or consumer use, enabling large-scale deployment and user adoption.

Data availability

Example images for training AI model are available at GitHub: https://github.com/szq1223/PHIND-system.git . If the source of images is unclear, their use for machine learning may be subject to ownership rights. Therefore, if additional training data are required, please ensure

that their collection complies with applicable local laws. All study participants provided informed consent, and the study protocol was approved by the relevant ethical committee (Stanford IRB Protocol no. 45621). All data were anonymized and handled according to strict privacy and data protection guidelines.

Code availability

All codes mentioned in this protocol are available at GitHub: https://github.com/szq1223/PHIND-system.git .

Received: 27 October 2024; Accepted: 7 October 2025;
Published online: 09 January 2026

References

1. Gambhir, S. S., Ge, T. J., Vermesh, O. & Spitler, R. Toward achieving precision health. *Sci. Transl. Med.* https://doi.org/10.1126/scitranslmed.aao3612 (2018).
2. Gambhir, S. S., Ge, T. J., Vermesh, O., Spitler, R. & Gold, G. E. Continuous health monitoring: an opportunity for precision health. *Sci. Transl. Med.* https://doi.org/10.1126/scitranslmed.abe5383 (2021).
3. Ge, T. J. et al. Passive monitoring by smart toilets for precision health. *Sci. Transl. Med.* **15**, eabk3489 (2023).
4. Park, S. M. et al. A mountable toilet system for personalized health monitoring via the analysis of excreta. *Nat. Biomed. Eng.* **4**, 624–635 (2020).
5. Melnik, A. V. et al. S-Wipe: stool sample collection for metabolomic gut health tracking. Preprint at *bioRXiv* https://doi.org/10.1101/2024.04.12.589313 (2024).
6. Foell, D., Wittkowski, H. & Roth, J. Monitoring disease activity by stool analyses: from occult blood to molecular markers of intestinal inflammation and damage. *Gut* **58**, 859–868 (2009).
7. Lewis, S. J. & Heaton, K. W. Stool form scale as a useful guide to intestinal transit time. *Scand. J. Gastroenterol.* **32**, 920–924 (1997).
8. Zhou, J. et al. Stool image analysis for digital health monitoring by smart toilets. *IEEE Internet Things J.* **10**, 3720–3734 (2022).
9. Halmos, E. P. et al. Inaccuracy of patient-reported descriptions of and satisfaction with bowel actions in irritable bowel syndrome. *Neurogastroenterol. Motil.* **30**, e13187 (2018).
10. Park, S. M., Ge, T. J., Won, D. D., Lee, J. K. & Liao, J. C. Digital biomarkers in human excreta. *Nat. Rev. Gastroenterol. Hepatol.* **18**, 521–522 (2021).
11. Rao, S. S. et al. S833 how valuable is an electronic stool diary app in the management of chronic idiopathic constipation (CIC) when using the vibrating capsule (VC). *Am. J. Gastroenterol.* **119**, S572–S573 (2024).
12. Lee, V. V. et al. Understanding the user: patients' perception, needs, and concerns of health apps for chronic constipation. *Digit. Health* **8**, 20552076221104673 (2022).
13. Litcher-Kelly, L., Kellerman, Q., Hanauer, S. B. & Stone, A. A. Feasibility and utility of an electronic diary to assess self-report symptoms in patients with inflammatory bowel disease. *Ann. Behav. Med.* **33**, 207–212 (2007).
14. Rogan, J., Bucci, S. & Firth, J. Health care professionals' views on the use of passive sensing, AI, and machine learning in mental health care: systematic review with meta-synthesis. *JMIR Ment. Health* **11**, e49577 (2024).
15. Park, S.-M., Hong, S., Joo, K., Kim, S. & Lepech, M. D. DigitalMe' in smart cities. *Innovation* **5**, 100678 (2024).
16. Li, S., Zhu, S. & Yu, J. The role of gut microbiota and metabolites in cancer chemotherapy. *J. Adv. Res.* **64**, 223–235 (2024).
17. Wang, Q., Su, M., Zhang, M. & Li, R. Integrating digital technologies and public health to fight COVID-19 pandemic: key technologies, applications, challenges and outlook of digital healthcare. *Int. J. Environ. Res. Public Health* **18**, 6053 (2021).
18. Ge, T. J., Chan, C. T., Lee, B. J., Liao, J. C. & Park, S.-M. Smart toilets for monitoring COVID-19 surges: passive diagnostics and public health. *npj Digit. Med.* **5**, 39 (2022).
19. Vandeputte, D. et al. Stool consistency is strongly associated with gut microbiota richness and composition, enterotypes and bacterial growth rates. *Gut* **65**, 57–62 (2016).
20. Procházková, N. et al. Advancing human gut microbiota research by considering gut transit time. *Gut* **72**, 180–191 (2023).
21. Self, M. M., Williams, A. E., Czyzewski, D. I., Weidler, E. M. & Shulman, R. J. Agreement between prospective diary data and retrospective questionnaire report of abdominal pain and stooling symptoms in children with irritable bowel syndrome. *Neurogastroenterol. Motil.* **27**, 1110–1119 (2015).
22. Russo, M. et al. Stool consistency, but not frequency, correlates with total gastrointestinal transit time in children. *J. Pediatr.* **162**, 1188–1192 (2013).
23. Saad, R. J. et al. Do stool form and frequency correlate with whole-gut and colonic transit? Results from a multicenter study in constipated individuals and healthy controls. *Am. J. Gastroenterol.* **105**, 403–411 (2010).
24. Wang, X. J. & Camilleri, M. A smart toilet for personalized health monitoring. *Nat. Rev. Gastroenterol. Hepatol.* **17**, 453–454 (2020).
25. Blake, M., Raker, J. & Whelan, K. Validity and reliability of the Bristol Stool Form Scale in healthy adults and patients with diarrhoea-predominant irritable bowel syndrome. *Aliment. Pharmacol. Ther.* **44**, 693–703 (2016).
26. Chumpitazi, B. P. et al. Bristol Stool Form Scale reliability and agreement decreases when determining Rome III stool form designations. *Neurogastroenterol. Motil.* **28**, 443–448 (2016).
27. Jaruvongvanich, V., Patcharatrakul, T. & Gonlachanvit, S. Prediction of delayed colonic transit using bristol stool form and stool frequency in eastern constipated patients: a difference from the west. *J. Neurogastroenterol. Motil.* **23**, 561 (2017).
28. Break the taboo with poo. *Nat. Rev. Gastroenterol. Hepatol.* **18**, 743–743 (2021).
29. We need to talk about crapping. *Nat. Microbiol.* **3**, 1189–1189 (2018).
30. Park, S.-M. Contrasting aspects of human excreta. *Nat. Rev. Gastroenterol. Hepatol.* **21**, 217–217 (2024).
31. Song, Z. et al. AI-driven defecation analysis by smart healthcare toilet: exploring biometric patterns and eu-tenesmus. *Adv. Sci.* **12**, 2503247 (2025).

Acknowledgements

This research was partially supported by a Start-Up Grant (SUG) and a Ministry of Education Tier 1 Grant (award number RS29/23) at Nanyang Technological University, Singapore. This work was additionally supported by the Tech Incubator Program for Startup Korea (TIPS) under grant number RS-2024-00458519. Additional funding was provided by the Stanford Maternal and Child Health Research Institute through the Stanford Medicine Children's Health Center for IBD and Celiac Disease. This work was also supported by the Agency for Science, Technology and Research (A*STAR) MTC Programmatic MedTech Thematic Grant Call 2024 (project ID: M24N9b0119). Partial support was also provided by the National Medical Research Council (NMRC), Singapore, under grant CNIG24jul-0013.

Author contributions

Z.S., M.K., J.L. and T.H.K. contributed equally to this work. They were primarily responsible for writing the manuscript, assembling the protocol and developing the code. J.K., J.H.K. and S.-G.K. performed independent verification and testing of the protocol. S.K. and J.S. provided administrative support, including materials, instruments, reagents, laboratory space and computing resources. B.-H.J., S.H.L., G.K., S.H.C. and N.M.-M. critically reviewed, commented on and edited the manuscript, with N.M.-M. focusing particularly on ethical considerations. W.G.P., I.S., M.J.R. and S.H.W. provided clinical oversight and mentorship throughout the project and during protocol development. C.W.O., X.S. and L.W.T.N. contributed essential computing resources. K.T.H.S., D.H. and S.W.S. provided key datasets necessary for machine learning training. K.C.Y. provided support in applying the PHIND system in the hospital setting. M.K. and H.-W.H. contributed to the hardware development. B.J.L. and S.-m.P. supervised the research and protocol development and provided overall project mentorship.

Competing interests

S.-m.P. is a cofounder of a stealth mode startup specializing in the implementation of a similar technology as that illustrated in the manuscript. D.H. is scientific cofounder and shareholder of KYAN Technologies. He is also a co-inventor of pending patents pertaining to AI-based drug development and personalized medicine. The other authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-025-01296-9.

Correspondence and requests for materials should be addressed to Brian J. Lee or Seung-min Park.

Peer review information *Nature Protocols* thanks Seungwan Seo, Youbin Zheng and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Zhiquan Song[1,16], Minso Kim[1,16], Jeung Lee[2,16], Tae Hyung Kwon[2,16], Juhwan Kim[1], Ji Hong Kim[1], Seong-Geon Kim[1], Soh Kim[2], Bong-Hyun Jun[3], Sang Hun Lee[4], Walter G. Park[5], Irene Sonu[5], Michael J. Rosen[6], Chi Wei Ong[1], Xiaotao Shen[1,7], Leonard Wei Tat Ng[8], Gun Kim[9], Sang Hoon Chae[8,10], Kewin Tien Ho Siah[11], Dean Ho[12], Shang Wei Song[12], Nicole Martinez-Martin[13], Juha Song[1], Kuo Chao Yew[14], Munho Kim[10], Hen-Wei Huang[7,10], Sunny H. Wong[7,14], Brian J. Lee[15] ✉ & Seung-min Park[1,7] ✉

[1]School of Chemistry, Chemical Engineering and Biotechnology, Nanyang Technological University, Singapore, Singapore. [2]Department of Civil and Environmental Engineering, Stanford University, Stanford, CA, USA. [3]Department of Bioscience and Biotechnology, Konkuk University, Seoul, Republic of Korea. [4]Department of Chemical and Biological Engineering, Hanbat National University, Daejeon, Republic of Korea. [5]Division of Gastroenterology and Hepatology, Department of Medicine, Stanford University School of Medicine, Stanford, CA, USA. [6]Division of Pediatric Gastroenterology, Hepatology and Nutrition, Department of Pediatrics, Stanford University School of Medicine, Stanford, CA, USA. [7]Lee Kong Chian School of Medicine, Nanyang Technological University, Singapore, Singapore. [8]School of Materials Science and Engineering, Nanyang Technological University, Singapore, Singapore. [9]School of Civil and Environmental Engineering, Nanyang Technological University, Singapore, Singapore. [10]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore. [11]Division of Gastroenterology and Hepatology, University Medicine Cluster, National University Hospital, Singapore, Singapore. [12]Department of Biomedical Engineering, College of Design and Engineering, National University of Singapore, Singapore, Singapore. [13]Center for Biomedical Ethics, Stanford University, Stanford, CA, USA. [14]Department of Gastroenterology and Hepatology, Tan Tock Seng Hospital, Singapore, Singapore. [15]School of Mechanical Engineering, Sungkyunkwan University, Suwon, Republic of Korea. [16]These authors contributed equally: Zhiquan Song, Minso Kim, Jeung Lee, Tae Hyung Kwon.