Journal of Electronic Research and Application, 2024, Volume 8, Issue 3
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Multi-Plate Microbial Monitoring Terminal Based on Raspberry Pi 4B

Qirong Luo, Xichang Cai*, Tongyuan Liu

School of Information, North China University of Technology, Beijing 100144, China

Corresponding author: Xichang Cai, caixc_ip@126.com

Abstract: We utilized Raspberry Pi 4B to develop a microbial monitoring system to simplify the microbial image-capturing process and facilitate the informatization of microbial observation results. The Raspberry Pi 4B firmware, developed under Python on the Linux platform, achieves sum verification of serial data, file upload based on TCP protocol, control of sequence light source and light valve, real-time self-test based on multithreading, and an experiment-oriented file management method. The system demonstrated improved code logic, scheduling, exception handling, and code readability.

Keyword: Raspberry Pi 4B; Object-oriented; Multithreading; Serial port protocol and parsing; TCP

Online publication: June 14, 2024

1. Introduction

Traditional microbial observation typically relies on optical microscopy, which is plagued by issues such as significant susceptibility to human factors, limited stability of observation results, and high reliance on the operator's experience. Furthermore, visual observation methods hinder the informatization of microbial observation results and pose challenges for file retention. These limitations underscore the need for some alternative approaches in the field of microbial observation.

To achieve the informatization and simplification of microbial imaging, Yin *et al.* developed a portable single-plate imaging system [1], while Zhang opted for wireless server access for data storage [2]. However, when confronted with large-scale experiments in the laboratory, the aforementioned solution still presents limitations such as inadequate experimental capacity and an unreliable wireless network.

It is imperative to employ embedded systems to streamline and enhance microbial monitoring experiments. Consequently, this project utilizes Raspberry Pi 4B to develop a microbial imaging system. The system controls via serial port for convenient management and storage, incorporates self-checking functionality, and can transmit images to the host computer using TCP protocol.

Shi's [3] Principle of Serial Communication provides theoretical support for the transmission reception, and handling of exceptions in this design. Chen [4] delineates various issues and their corresponding solutions that may arise during the fungal shooting process, while Yu offers guidance for utilizing STM32 as the control ter-

minal and implementing a Raspberry Pi controlled by a serial port [5]. Cao provides a reference for the realization of the whole system [6].

2. Overall structure

This project aims to optimize and digitize the process of microbial monitoring, utilizing Raspberry Pi 4B for image capturing. The implementation of the terminal capturing is divided into three primary components: self-testing and mutual testing, capturing task, and follow-up of failed experiments.

The peripheral circuit of Raspberry Pi 4B consists of a sequence light source, power module, PC upper computer, STM32F4, and Raspberry Pi HQ Camera. The hardware block diagram is shown in **Figure 1**.

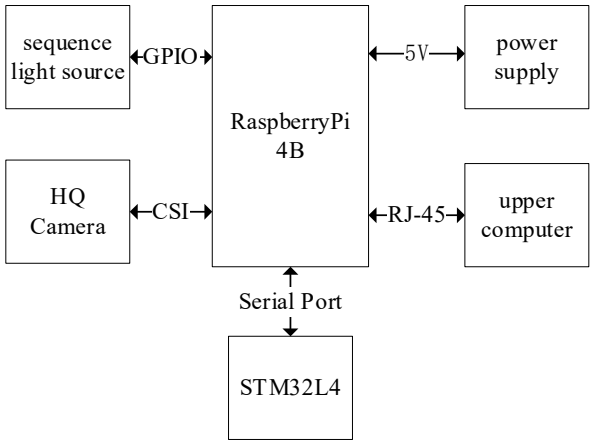

Figure 1. Hardware block diagram

The workflow of the Raspberry Pi 4B encompasses subtasks such as self-testing, experiment configuration, image capturing, and uploading. The program ensures that all image-capturing tasks are directly related to the experiment, with the experiment serving as the minimum execution unit of the program. The procedural flow is shown in **Figure 2**.

In the self-test phase, the system initializes the log recorder, sequence light source, and light valve before conducting the camera self-test, the serial port mutual test, and obtaining the operating parameters. During the experiment setup stage, experimental information must be set priorly. Upon receiving correct instructions, an experiment folder will be created; An error message is returned if the instruction is not to set the experiment information. During the capturing stage, the previously generated experimental path will be used as the photo path. Real-time monitoring of hardware resources is conducted through multithreading during folder path creation and image capturing. Once capturing is completed, pictures are uploaded via TCP based on user-selected mode. Any errors encountered when uploading will be sent back and recorded.

The program conducts self-tests on the hardware upon system startup to ensure stable operation. Experimental parameters are established prior to capturing the task in order to safeguard data integrity. The system uploads data according to user-selected modes, providing flexibility. Real-time resource monitoring allows for timely problem detection and enhances system maintainability. Overall, the design emphasizes system flexibility while ensuring stability and data security, resulting in a reliable and efficient system.

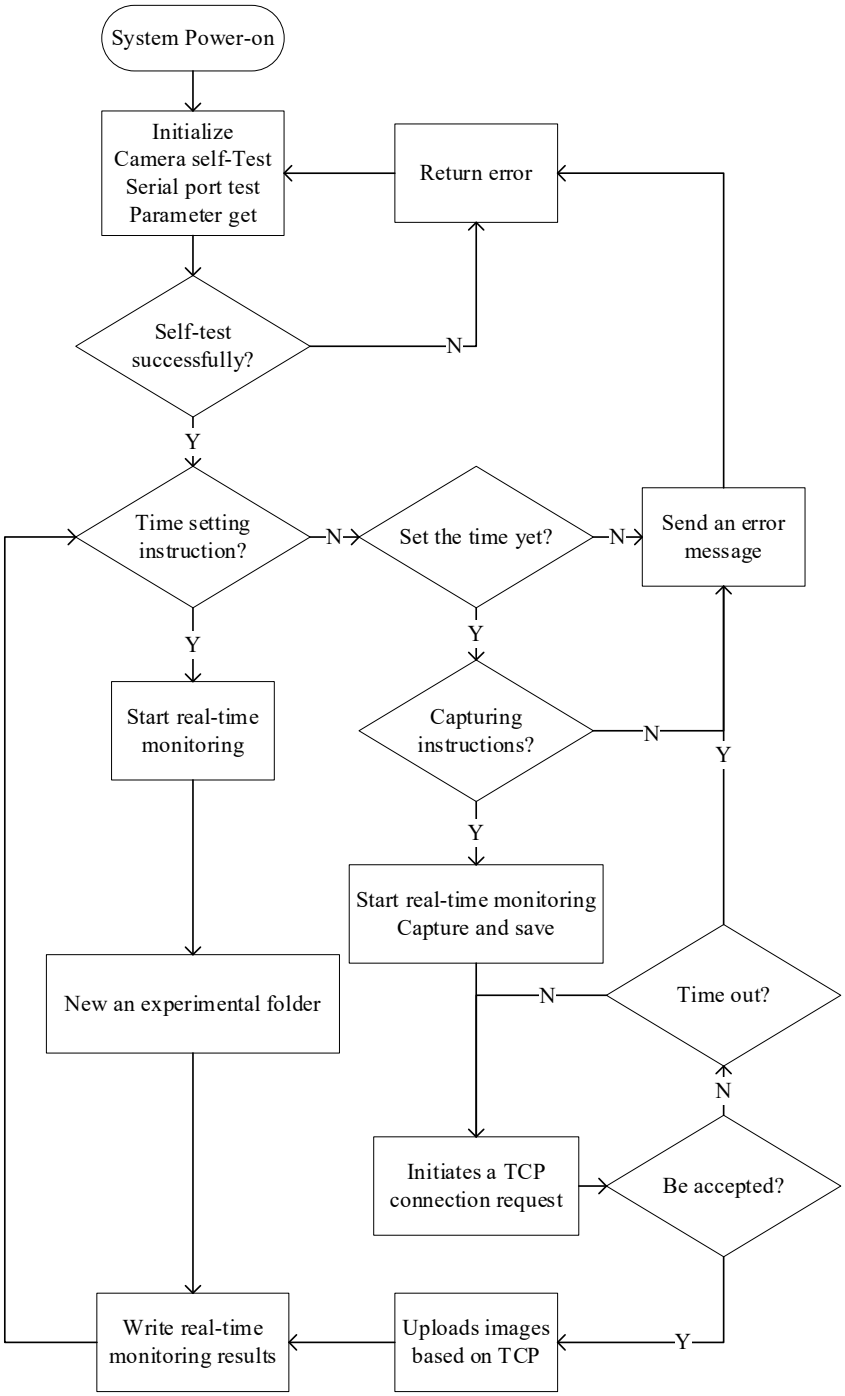

Figure 2. Program flow

3. Details of the design

3.1. Object-oriented Implementation

In consideration of the usage frequency of the camera and serial port, this design objectifies these two types of hardware to enhance the ease of use and maintainability of the code. The utilization of object-oriented design facilitates the construction of code that is highly modular and easily extensible. Furthermore, to further enhance code readability and facilitate code management, this design also renders the experiment object-oriented, thereby enabling a clearer organization and management structure for the code.

In terms of Camera, this design derives the Camera class based on the PiCamera class to better manage the

camera function. The Camera class contains sub-attributes such as the photo path, which requires that the path should be set before capturing to ensure the photo can be saved correctly. In addition, according to the different display requirements of different strains, the design also takes the configuration of sequence light source and light valve as the priority setting item of capturing task. At the same time, the Camera class provides a method to manually set the camera exposure factors and white balance parameters, which is used to adjust the display effect of the picture. Lastly, the method interface of the quick capturing mode is implemented to improve the shooting efficiency.

On the serial port side, this design derives the UART class based on Serial class to better handle serial port data. The UART class defines the structure of received serial port data at the Raspberry Pi 4B end, encompassing frame header, device ID, command, data length, data content, checksum, and frame tail. This specification facilitates subsequent processing by enabling efficient discrimination and segmentation of data through the UART class methods.

This design also makes the experiment object-oriented in order to better organize and manage the information and operations related to the experiment. The Raspberry Pi 4B will instantiate the corresponding experimental object after correctly parsing the experimental information setting instructions. The Experiment class contains a method for establishing folder paths, which is used to organize folder paths based on experiments. The Camera class is defined as a sub-attribute of the Experiment class, so that the capturing process can be executed normally after the successful instantiation of the experiment object and configuration of relevant information, thereby improving the stability of the capturing work process. The organization of each class is shown in **Figure 3**.

Through the utilization of object-oriented design, this project facilitates code maintenance and debugging, while also improving code readability and extensibility.

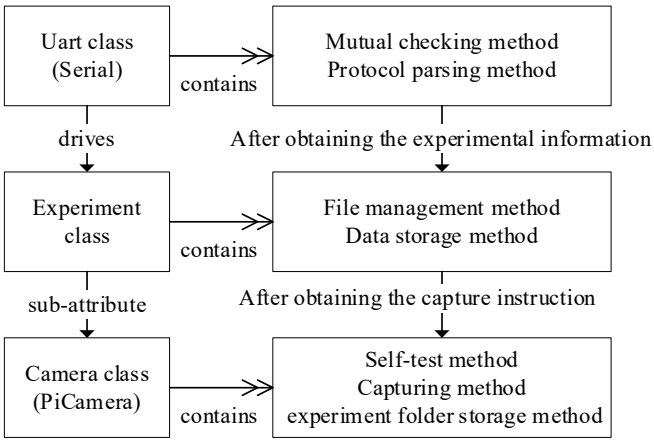

Figure 3. The organization of the classes

3.2. Real-time monitoring

The Raspberry Pi 4B, as a lightweight Linux platform, necessitates ongoing resource monitoring to guarantee adequate hardware resources and prevent data errors stemming from insufficient resources when accessing high-performance peripheral devices such as cameras. This real-time monitoring is essential for ensuring system stability.

When the Raspberry Pi 4B is capturing photos and writing files, a new thread will be initiated to conduct real-time monitoring of the resources of the Raspberry Pi 4B. This monitoring thread utilizes an infinite loop to continuously acquire the parameters of the resource. Upon completion of the thread, the maximum value

is selected as a benchmark for performance requirements to determine whether the resource limit has been exceeded. A mutex must be attached to the data within the monitoring thread to prevent data inconsistencies resulting from kernel conflicts. Furthermore, the queue parameter transmission method is employed for obtaining and storing self-test result data generated during the real-time monitoring process, ensuring the integrity and accuracy of the data.

3.3. Acquisition of operation parameters

This function is developed based on the implementation of instructions within "PiOS", utilizing commands such as "cat" and "vcgencmd" to retrieve system performance parameters. By analyzing these parameters, potential hardware issues can be promptly identified. Upon detection of any problem, the system will promptly transmit the information back to STM32 in order to ensure the stability and reliability of the hardware environment.

The performance parameters acquisition of the Raspberry Pi 4B are crucial for self-testing and real-time monitoring. As a result, this design integrates methods for both self-testing and real-time monitoring to create a unified interface, thereby reducing code redundancy. This approach enhances code clarity and conciseness while also improving code maintainability and expandability, laying a foundation for the stable operation of the hardware system.

3.4. Uploading

The TCP upload function is designed to accommodate various demand scenarios by offering initiative and passive upload modes. In the initiative mode, upon completion of an experiment, the Raspberry Pi 4B initiates a TCP connection request to the upper computer. Failure of the upper computer to respond within a specified time period results in marking the experiment and sending it back as an error message. This approach ensures timely uploading of experimental data while also enabling prompt identification and handling of abnormal uploads, thereby ensuring system stability and reliability.

In passive upload mode, the Raspberry Pi 4B initiates a TCP connection request to the upper computer only after receipt of the corresponding control command. This operational configuration enables the Raspberry Pi 4B to await external commands for triggering upload actions, thereby facilitating a more adaptable upload mechanism. Consequently, users can exercise control over data upload timing in accordance with specific circumstances, thus meeting diverse data transmission requirements and enhancing system customizability and applicability.

The system's two upload modes are designed to dynamically select the most appropriate mode based on the specific circumstances, thereby enhancing the system's flexibility and adaptability. Additionally, timely response and resolution of abnormal uploading situations ensure the integrity and accuracy of experimental data, thus providing a robust guarantee for the smooth operation of the system.

3.5. Protocol analysis and workflow

The whole process of Raspberry Pi 4B capturing task needs to be completed under the serial port data control from STM32. Raspberry Pi 4B requires the received serial data to be consistent with the workflow, otherwise, it will send back an error alarm message. Multiple capturing and uploading can be carried out under the same experiment, but the capturing task cannot be completed without setting experimental parameters.

The data bit in the serial port protocol contains experimental information, which must be analyzed systematically based on the corresponding relationship between the two parties. This data serves as fundamental information for the Experimental class and should be configured as a priority. To facilitate rapid capturing, a specific method is provided within the Camera class, enabling the Raspberry Pi 4B to capture photos

while artificially concealing certain experimental parameters. However, this mode does not establish a file management path based on experiments and is only suitable for scenarios that prioritize capturing frequency.

4. Summary and development

4.1. Summary

To establish a dependable microbial monitoring capturing system, the design significantly enhances chip performance scheduling, experimental results presentation, and information post-processing. The code is both readable and extensible, thus capable of meeting increasingly advanced future needs.

4.2. Development

The program currently lacks a preview feature for capturing. In the experiment, individuals tend to manually adjust the desired effect prior to capturing an image. The system does not currently support capturing previews. To address this limitation, we plan to utilize the "PyQt" method based on multi-threading to develop a capturing preview function on Raspberry Pi 4B in future research.

Author contributions

Conceptualization: Qirong Luo
Writing: Qirong Luo and Tongyuan Liu
Technical support: Xichang Cai

Disclosure statement

The authors declare no conflict of interest.

References

[1] Yin D, Cai X, Chen J, et al., 2023, Design of Single Plate Microbiological Monitoring System for Scientific Research. China High-Tech, 14: 104–106 + 118. https://www.doi.org/10.13535/j.cnki.10-1507/n.2023.14.30

[2] Zhang J, Li Y, Zhang Z, et al., 2023, Design and implementation of Raspberry Pi Intelligent Flower Watering System. Computer Knowledge and Technology, 19(19): 107–109 + 116. https://www.doi.org/10.14004/j.cnki.ckt.2023.1091

[3] Shi J, 2022, In-depth Exploration of Common Serial Data Asynchronous Communication Protocols. Microcontrollers & Embedded Systems.2022(7): 26–29.

[4] Chen D, Chen C, 2017, Discussion on the Application of Microbiological Picture Shooting Technology. Chin J Clin Lab Sci. 35(10): 729–735. https://www.doi.org/10.13602/j.cnki.jcls.2017.10.03

[5] Yu Q, Guan Y, Huang W, et al., 2023, Research on Aquaculture Water Quality Monitoring Unmanned Boat System Based on STM32 and Raspberry Pi. Fishery Modernization. 50(5): 33–42.

[6] Cao X, 2021, Design and Implementation of Pathogenic Microorganism Sample Processing and Incubation Photographing System, thesis, Xidian University.