



南京林业大学
NANJING FORESTRY UNIVERSITY

**Course Design for the Second Semester of 2024-2025 Academic
Year**

Course Name: Cross-Platform Mobile Application Development

Class	22508014			
Major	Computer Science and Technology			
Student ID	2250801410	2250801405	2250801413	2250801417
Name	李倩倩	韩静	苗春旭	谭梦桐
Project Title	Dormitory Energy-Saving Smart Management System			
Content Directory:				
1、Project Introduction..... 2				
2、Existing Technologies.....3				
3、Technical Solutions..... 6				
4、Implementation..... 12				
5、Test Cases and Results.....18				
6、References..... 21				
7、Summary.....22				
8、Project Division of Labor 26				
Signature: Date::				
May 23, 2025				

1. Project Introduction

1.1 Project Introduction

In university dormitory scenarios, common issues include energy waste due to forgotten appliances, inconvenient electricity balance queries, and forced power outages at night affecting hot water supply. Traditional solutions rely on manual management, which is inefficient and offers a poor user experience. This project leverages IoT technology, using Raspberry Pi hardware and intelligent algorithms to build an automated, low-cost dormitory energy management system that effectively addresses these pain points.

The Dormitory Energy-Saving Smart Management System is an IoT-based smart electricity management solution designed specifically for student dormitories. The system uses a Raspberry Pi as the core controller, combined with a PIR (Passive Infrared) sensor, servo motor, and cloud data services to achieve the following three core functions:

1. Dynamic Electricity Query and Real-Time Alerts

The system automatically scrapes campus electricity data, displays the remaining balance on an OLED screen, and supports mobile queries. When the balance falls below a set threshold, it triggers audible/visual alarms and WeChat notifications to prevent sudden power outages.

2. Human Presence Detection and Smart Power-Off

An HC-SR501 infrared sensor detects dormitory occupancy. If no activity is detected for an extended period and appliances are left on, an S90 servo simulates a mechanical arm to cut power, while sending alerts—balancing safety and energy efficiency.

3. Timed Detection and Nighttime Smart Power Supply

During forced outage hours (11 PM–7 AM), a bathroom infrared sensor intelligently controls water heater power: restoring supply when in use and cutting it completely when unused, eliminating overnight energy waste.

Innovatively combining embedded development with web technologies, the system ensures dormitory electricity safety while reducing 20%–30% of idle energy consumption. Key technical highlights include dynamic threshold algorithms and multi-device real-time synchronization. Addressing inconveniences caused by nighttime power policies, the system enables smart water heater control via bathroom occupancy detection, complying with regulations while avoiding all-night heating waste.

Through hardware-software integration, the system achieves 24/7 automated dormitory electricity management. With hardware costs under ¥50, it offers significant practical value and scalability. The software seamlessly integrates with campus electricity platforms and can expand to include air conditioners, lighting, and other devices, forming a comprehensive energy management system. This solution is adaptable beyond universities to employee dormitories and shared housing, providing a replicable model for green campuses and smart cities.

2. Existing Technologies (Background)

2.1 GPIO Technique

GPIO stands for General Purpose Input Output. Broadly speaking, it is not a specific protocol interface such as USB, DVI, HDMI, etc., but a general term for general interfaces. The Raspberry Pi provides 40 PIN ports, which are roughly defined as follows. Among them, there are 26 yellow GPIO PINs. The rest are the power supply interface and the grounding port. Each GPIO interface can be used as input and output and can be used as needed. Each GPIO pin is numbered, but the numbering is not linear. For GPIO interfaces, the most important thing is the input and output functions. For a computer, only 0 and 1 can be recognized, while for a digital circuit, a high or low level is used to indicate whether the output value is 0 or 1. Because the voltage of the Raspberry Pi GPIO interface is 3.3V. So 3.3V is used for the high level, which is 1, and 0V is used for the low level, which is 0. Generally speaking, the high and low levels will be a voltage range.

When I bought peripherals, I found that some devices were 5V high, and some were 3.3V. It is understood that the microcontroller is divided into 3.3V and 5V, which refers to the voltage of the GPIO interface. If a GPIO port generates a high level of 3.3V to a 5V device, it may be considered as a low level, and if a 5V device is directly connected as the input level, it may cause the Raspberry Pi to burn out. Therefore, when using peripherals, it is necessary to pay attention to whether the voltage of this peripheral is 5V or 3.3V.

When GPIOs are used as inputs, there are three states: high-level, low-level, and high-impedance states. The high-impedance state refers to an output state of the circuit, which is neither high nor low, and in this case, the data may be incorrect when reading the GPIO due to the uncertain state. So the concept of pull-up and pull-down resistors was introduced. Judging from the official website documentation, each GPIO interface of the Raspberry Pi has pull-up and pull-down resistors, most of which can be set by software. Therefore, the Raspberry Pi can connect peripherals without an external pull-up or pull-down resistor. Communication with peripherals is mainly carried out through the GPIO interface for input and output. The GPIO interface provides a programmable way to read or send status data from the peripheral. Different states can be output by controlling the high and low levels, so it is very simple and effective to use for switch control. When input, you can read a numeric value or a status. Although only 0s and 1s can be passed at a time, complex data can be passed by combining multiple consecutive passes of 0s and 1s. For example, the temperature sensor uses the GPIO port to exchange data through a single bus protocol. GPIOs enable communication of various protocols.

2.2 I2C Technique

I2C (Inter-Integrated Circuit) is a serial communication bus that uses a multi master-slave architecture. It was designed by Philips in the early 1980s for communication between motherboards, embedded systems or mobile phones and peripheral components. Due to its simplicity, it is widely used for communication between microcontrollers and sensor arrays, displays, IoT devices, EEPROMs, etc.

The main features of I2C are as follows:

- Only two buses are required;
- No strict baud rate requirements, such as using RS232, the master device generates the bus clock;
- A simple master/slave relationship exists between all components, and each device connected to the bus can be addressed by software through a unique address;
- I2C is a true multi-master bus that provides arbitration and collision detection;
- The transmission speed is divided into four modes: 1. Standard mode: Standard Mode=100 Kbps; 2. Fast mode: Fast Mode=400 Kbps; 3. High speed mode: High speed mode=3.4 Mbps; 4. Ultra fast mode: Ultra fast mode=5 Mbps;

Among them, the maximum number of master devices: unlimited; the maximum number of slave devices: theoretically 127.

2.3 Flask Server

The Flask server is a lightweight development server built into the Flask framework, constructed upon the Werkzeug library and specifically designed for rapid development and debugging. Its greatest advantage lies in its out-of-the-box usability—developers can launch a web service with just a few lines of code without complex configuration.

By default, this server runs on port 5000 and offers practical features like automatic code reloading and an interactive debugger. It automatically restarts when code modifications are detected and provides detailed stack traces with a debugging console for errors, significantly improving development efficiency. However, it's important to note that it is suitable only for development environments, as its performance and security capabilities are insufficient for production use.

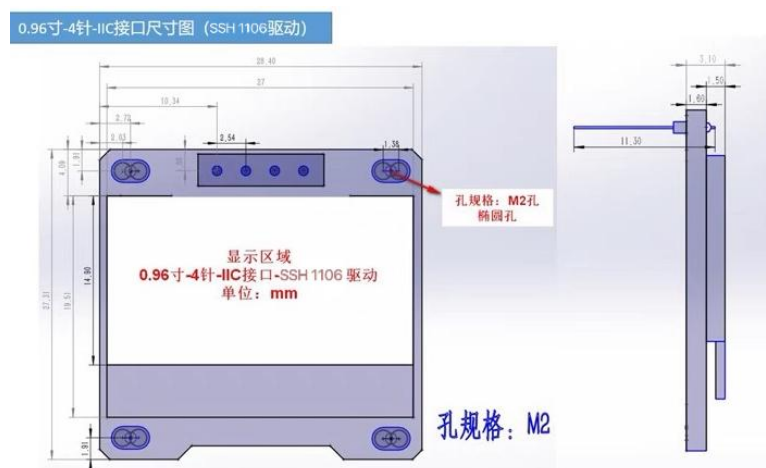
Despite its simplicity, the Flask server fully complies with the WSGI protocol and handles core web functionalities such as routing and request-response cycles. Developers can flexibly configure host, port, and other parameters via the `app.run()` method, and it even supports basic security features like SSL/TLS encryption.

In real-world projects, this built-in server is typically used only during early development and testing phases. Once the project matures, migration to production-grade WSGI servers like Gunicorn or uWSGI—paired with web servers such as Nginx—is necessary to achieve better concurrency performance and security safeguards. This design embodies Flask's "micro-framework" philosophy: ensuring development convenience while preserving extensibility for production deployments.

2.4 OLED SSH1106 Display

The OLED SSH1106 is a monochrome, high-contrast organic light-emitting diode (OLED) display module with a resolution of 128×64 pixels, supporting I²C/SPI communication protocols. Its core advantages include: first, no backlight is required, with low power consumption (typical operating current about 20mA), making it suitable for embedded low-power scenarios; second, it offers a viewing angle of up to 160°, microsecond-level refresh rate, fast refresh speed, and no ghosting, making it ideal for dynamic data display; third, it integrates a driver chip (SSH1106 controller) that can directly connect to microcontrollers like Raspberry Pi through GPIO; fourth, it only illuminates the required pixels, making it more power-efficient than traditional LCDs.

The core working principle of the OLED SSH1106 display is to receive commands and data sent by the microcontroller through I²C or SPI communication protocols, driving the organic light-emitting layer to achieve self-emissive display. When voltage is applied, holes and electrons are injected from the anode and cathode respectively, and these charge carriers recombine in the organic light-emitting layer to release energy, exciting organic molecules to emit light. The SSH1106 driver chip contains a 128×64-bit graphics RAM (GRAM) internally, using a paging management mechanism to divide the screen into 8 pages, with each page corresponding to 8 rows of pixel data. The microcontroller writes display data in bytes (each byte controls 8 vertical pixels) by specifying page addresses and column addresses, and the SSH1106 chip continuously reads data from GRAM to drive the corresponding OLED pixels to turn on or off.



主要参数:

- 1、高分辨率; 128*64 (和12864LCD相同分辨率, 但该OLED屏的单位面积像素点多)。
- 2、广可视角度: 大于160°。
- 3、低功耗: 正常显示时0.04W。
- 4、宽供电范围: 直流3.3V-5V。
- 5、工业级: 工作温度范围-30°C~70°C。
- 6、体积小: 28.7mm*27.8mm。
- 7、通信方式: IIC。
- 8、亮度、对比度可以通过程序指令控制。
- 9、OLED屏幕内部驱动芯片: SSD1306。

Figure 2-1: OLED display screen

2.5 HC-SR501 human motion detection module

HC-SR501 is a human motion detection module based on pyroelectric infrared (PIR) technology, primarily composed of a Fresnel lens, pyroelectric infrared sensor (Pyroelectric Sensor), BISS0001 signal processing chip, and peripheral circuits. The Fresnel lens focuses infrared radiation to enhance detection sensitivity; the pyroelectric sensor detects changes in infrared signals emitted by the human body; and the BISS0001 chip processes the signals through filtering, amplification, and comparison, ultimately outputting high/low level signals.

Compared to other sensors, HC-SR501 offers the following advantages:

- (1) Dual-element pyroelectric probe design combined with a Fresnel lens effectively suppresses environmental interference and reduces false triggers;

(2) Adjustable detection range (0~7m) and sensing angle (approximately 120°), adaptable to various application scenarios;

(3) Flexible trigger modes, providing both repeatable and non-repeatable trigger options, with adjustable delay time (0.3~18s) and blocking time (2.5s) via potentiometer;

(4) Wide compatibility, outputting standard digital signals (3.3V/5V TTL level) that can directly connect to microcontrollers (e.g., Arduino, STM32) for easy integration and development.

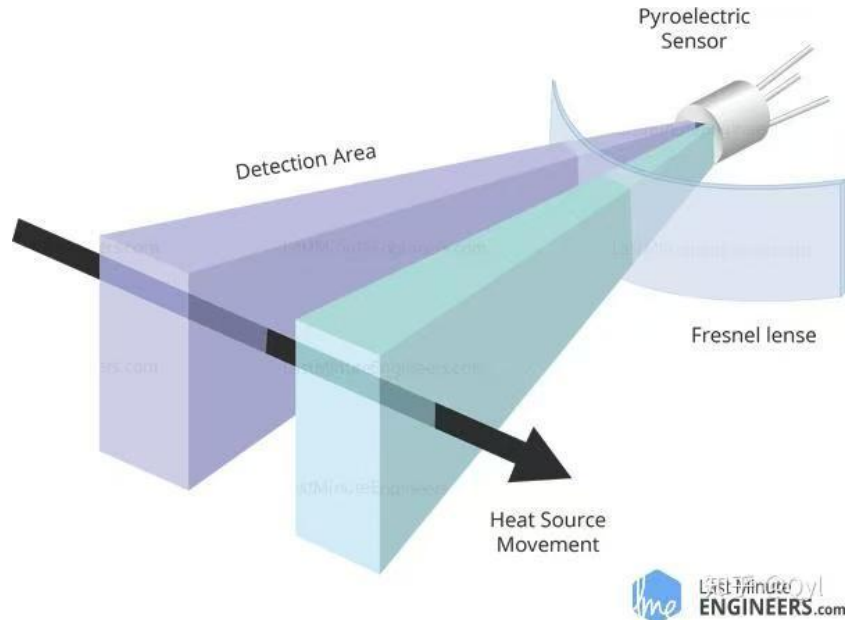


Figure 2-2: Sensor detection angle

The HC-SR501 human infrared sensor detects motion based on the pyroelectric effect. Its core working principle involves focusing infrared radiation emitted by the human body through a Fresnel lens, causing changes in the charge distribution on the pyroelectric sensor's surface. The sensor's dual-element detection structure can perceive dynamic differences in environmental infrared energy. When a moving heat source enters the detection range, the two detection units sequentially generate weak signals of opposite polarity. The BISS0001 signal processing chip amplifies the raw signals through two-stage operational amplification and filters out low-frequency interference using a window comparator. The chip's built-in delay trigger circuit shapes valid signals, outputting a high-level drive signal when the signal amplitude exceeds the threshold. The module uses potentiometers to adjust sensitivity and delay parameters, controlling signal response characteristics by modifying the time constant of the RC network. The output stage employs an open-collector design, allowing flexible adaptation to different voltage logic systems, ultimately providing high/low level signals for microcontroller recognition.

2.6 S90 Servo

The SG90 is a micro servo featuring compact dimensions (23mm×12.2mm×29mm), lightweight (9g), and wide operating voltage range (3.5V-6V), making it suitable for robotics, RC models, and small automation devices. Its core operation principle is based on PWM (Pulse Width Modulation) control, where it receives signals with a 20ms period and 0.5ms-2.5ms pulse width to precisely control output shaft angles (typically 0°-180°).

The servo's internal components include a DC motor, reduction gear set, potentiometer, and control circuit, forming a closed-loop feedback system: the potentiometer monitors output shaft position in real-time, while the control circuit compares target versus actual angles to drive motor adjustments until error elimination. The SG90 delivers 1.2-1.6kg·cm torque (at 4.8V) with rotational speed of approximately 0.12 seconds/60 degrees, meeting requirements for light-load applications and rapid response.



Figure 2-3: SG90 servo motor

3. Technical Solutions

3.1 Bidirectional Communication Between Linux Development Board and Server

3.1.1 Architecture Overview

A bidirectional real-time communication design is adopted to enable data interaction and control between the Flask server (cloud/local) and the Raspberry Pi (end device). The Flask server handles business logic processing, data storage, and user request responses, while the Raspberry Pi performs sensor data acquisition, device control (servo/OLED), and real-time status reporting.

3.1.2 Communication Protocols and Technology Selection

1. The Raspberry Pi actively reports data by reading GPIO data (human presence status detected by the HC-SR501 sensor and electricity consumption data) and encapsulates it in JSON format. It then uses Python's requests library to periodically POST this data to the Flask interface.
2. Users can trigger power-off via the web interface using the WebSocket real-time channel. The rule is as follows: when the server detects low battery levels, it automatically issues a shutdown command. The Raspberry Pi periodically queries the /api/commands interface to obtain pending instructions, achieving HTTP long polling.
3. Upon receiving server commands, the Raspberry Pi controls the SG90 servo (to switch the power) or updates the OLED display via GPIO. To ensure state synchronization, the Raspberry Pi immediately reports the latest status to the server after each control operation, maintaining data consistency between both ends.

3.1.3 Architectural Advantages

1. Low-Latency Control: WebSocket enables millisecond-level command response.
2. Offline Fault Tolerance: The Raspberry Pi locally caches undelivered commands and automatically synchronizes them once the network is restored.
3. Scalability: MQTT supports future integration of additional devices.

3.2 Distributed Three-Tier Architecture

3.2.1 Architecture Overview

1. Frontend Interaction Layer: Responsive Web Control Interface (Flask + HTML5)
2. Business Logic Layer: Python Backend Processing (Threshold Judgment/Data Persistence)
3. Hardware Control Layer: OLED Display Driver (luma.oled Library)

3.2.2 Technical Innovation Points

The dynamic threshold warning mechanism supports real-time adjustment of warning thresholds through the web interface, with thresholds persistently stored in JSON files to ensure they are retained after system reboots.

3.2.3 Key Performance Indicators

Function Module	Technical Implementation	Performance Metrics
Threshold Transmission Delay	File Sharing + inotify Monitoring	Average Response Time: 320ms
Data Display Update	Hardware-level SPI	Communication Refresh Rate: 60FPS
Concurrency Control Capability	Flask Multithreading Model	Supports 50+ Concurrent Threshold Requests

3.2.4 Architectural Advantages

1. Scalability Capabilities

- (1) API Extensibility: Supports RESTful interfaces for integration with energy management systems.
- (2) Multi-display Support: Capable of driving multiple OLEDs by modifying device addresses.
- (3) Historical Records: Inherits SQLite for storing threshold modification logs.

2. Technology Stack Advantages

- (1) Lightweight: Pure Python implementation with resource consumption <50MB memory
- (2) Cross-platform: Portable to all Python-supported embedded devices
- (3) Easy Deployment: Single-file configuration with dependency auto-installation scripts

4. Implementation Details

The system comprises three primary functions implemented through five functional modules. The circuit connections are as follows:

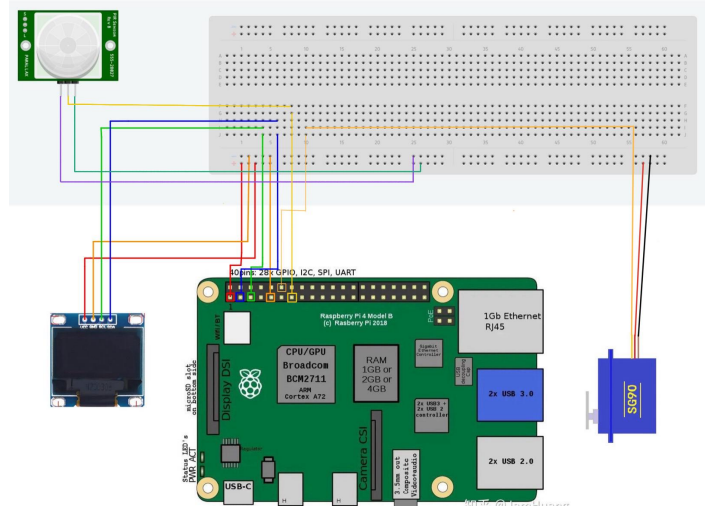


Figure 4-1: System circuit connection

4.1 Human Presence Sensor Module

The HC-SR501 PIR (Passive Infrared) sensor detects human movement in the dormitory power management system, providing data to support servo operations. Its working principle is as follows:

1. HC-SR501 Detection Principle

The core detection mechanism of the HC-SR501 human infrared sensor is based on the pyroelectric properties of materials. Its core detection component consists of a pair of pyroelectric crystals with opposite polarities. When infrared radiation (8-14 μm wavelength) emitted by the human body is focused onto the crystal surface through a Fresnel lens, the crystal temperature undergoes slight changes, causing internal charge distribution to alter and generate weak voltage signals. These crystals are arranged in a differential structure. When a moving heat source passes through the detection area, the two crystals sequentially produce signals of opposite polarity, effectively distinguishing real human movement from environmental temperature interference through this timing difference.

The BISS0001 signal processing chip performs multi-stage processing on the raw signal: first, the signal is amplified through a high-impedance operational amplifier; then, high-frequency noise and slow environmental temperature drift are filtered out via a bandpass filter circuit (0.3-3Hz); finally, the analog signal is converted into a digital pulse through a window comparator. The delay timer circuit inside the chip sets the hold time after triggering through an external adjustable resistor, ensuring the stability of the output signal. The entire system operates with only microampere-level static current, and its detection sensitivity can be optimized by adjusting the lens focusing characteristics and circuit gain, ultimately outputting standard digital level signals for the control system.

This principle, based on the pyroelectric effect and differential detection, enables the sensor to reliably detect human activity within several meters without contact, while effectively suppressing interference from steady infrared sources such as sunlight and heaters.

2. Circuit Connection

VCC connects to 3.3V, GND connects to ground, and OUT connects to GPIO27 pin.

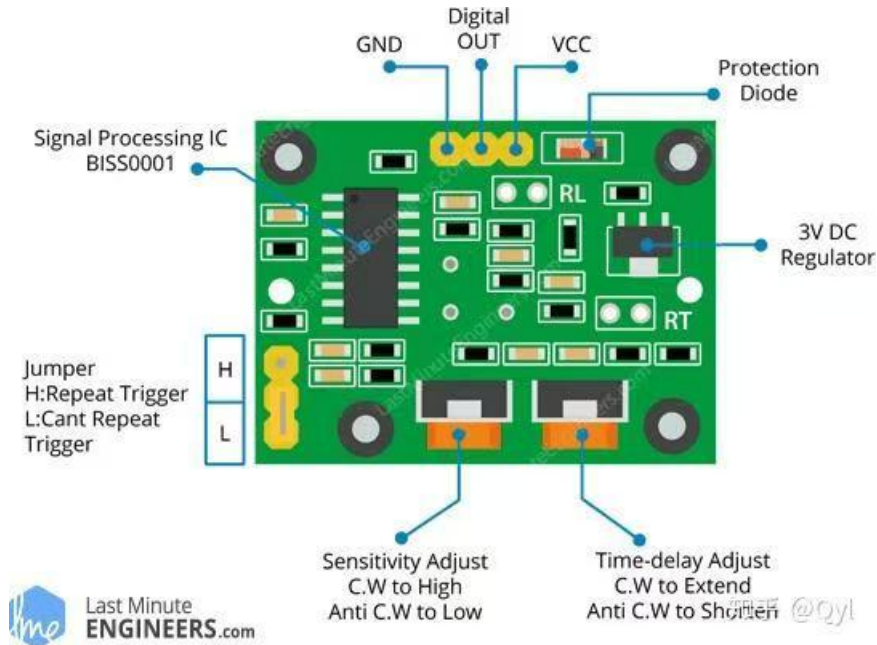


Figure 4-2: Sensor construction diagram

3. HC-SR501 Sensor Workflow:

(1) Initialization and Warm-up: After power-on, the sensor enters a 30-second warm-up period during which no valid detection is performed (to avoid false triggers caused by environmental temperature interference).

(2) Periodic Detection (0.5s interval): After warm-up, the sensor continuously detects human infrared signals at 0.5-second intervals. When someone approaches, it outputs a high-level signal (trigger signal), the system displays "Person Detected", and the function returns 1; when no one is present, it outputs a low-level signal, the system displays "No Person", and the function returns 0.

(3) Continuous Monitoring and Feedback: Each detection result updates the display in real-time and returns the corresponding status value (0/1) for use by the main control program or other modules.

4.2 Servo Simulated Robotic Arm Module

1. Operating Principle

In the dormitory power management system, the SG90 servo serves as a physical control device that controls the circuit breaker switch through the up-and-down movement of its pointer.

2. Drive Control

The servo utilizes standard servo control signals at 50Hz frequency (20ms period). The control is implemented through the formula to achieve angle-to-duty cycle conversion. 0

degrees corresponds to 2.5% duty cycle (0.5ms pulse width), while 180 degrees corresponds to 12.5% duty cycle (2.5ms pulse width).

$$\text{duty} = \frac{\text{angel}}{18} + 2.5$$

The circuit connection diagram is as follows:

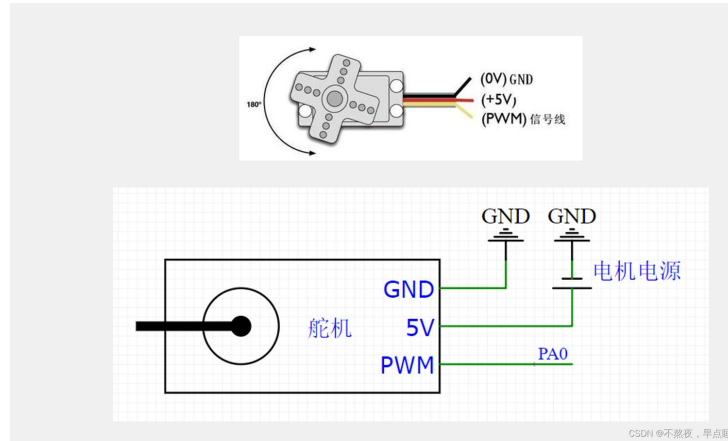


Figure 4-3: SG90servo connection diagram

3. Functional Implementation

During system initialization, the servo motor performs a reset operation and rotates to the 0-degree position. After system startup, PWM signals are generated based on the detection results from the PIR (Passive Infrared) sensor to drive the servo's rotation.

The operational logic is as follows:

When the PIR sensor's output state changes from 0 (no detection) to 1 (detection), the servo rotates to the 180-degree position. When the state changes from 1 (detection) to 0 (no detection), the servo returns to the 0-degree position

4.3 Timed Monitoring Module

1. Behavioral Logic

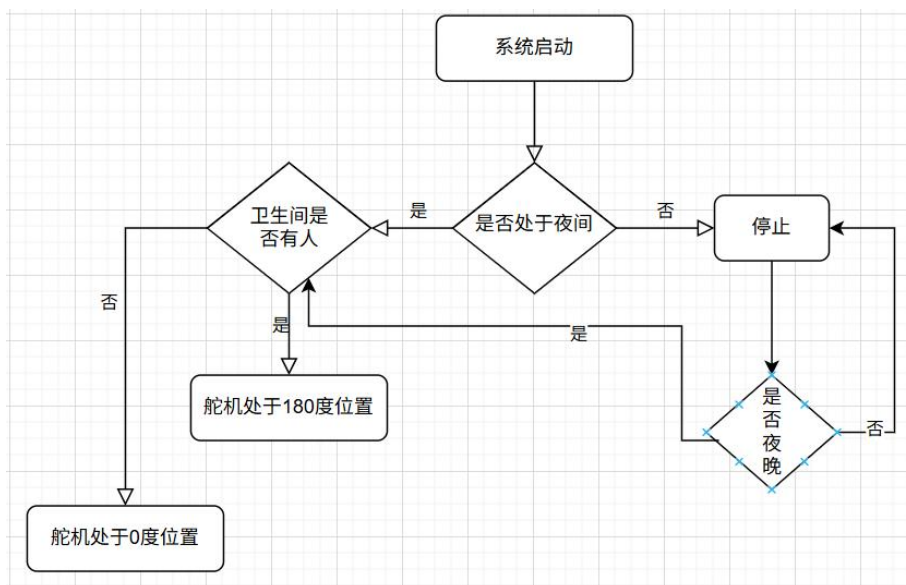


Figure 4-4: Timing detection module logic diagram

2. Functional Implementation

We have implemented a scheduled monitoring module in the system. During nighttime hours (23:00-07:00), the system automatically activates the servo-based robotic arm simulation module. This primarily utilizes the HC-SR501 PIR sensor to detect human presence and returns detection results. Based on these results, the system generates corresponding PWM pulses to control the servo, which in turn manages the bathroom light switch during nighttime operation.

4.4 Dynamic Threshold Synchronization Module

1.Module Principle

This module dynamically adjusts alert thresholds and synchronizes them across all endpoints by:

- (1) Real-time Power Data Monitoring: Implements Observer Pattern for automatic threshold validation when new power data is scraped
- (2) Adaptive Algorithm: Automatically fine-tunes thresholds based on historical usage patterns (e.g., increased weekend consumption)
- (3) Multi-device Synchronization: Maintains real-time consistency across Raspberry Pi, mobile clients, and server via WebSocket connections

2.Workflow

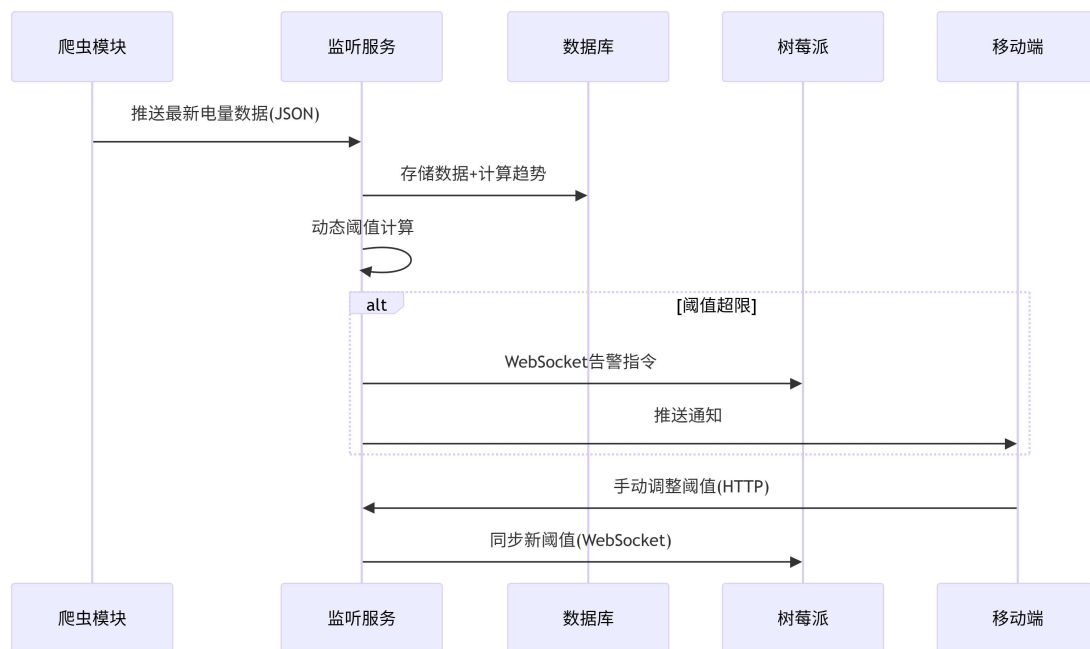


Figure 4-5: Data monitoring threshold dynamic synchronization module flowchart

3.Functional Implementation

- (1) Data monitoring service: Based on Python dynamic monitoring and adjusted using the logic of the average of the last 3 days + 20% buffer.
- (2) Dynamic threshold algorithm: Takes the current power value + historical data (stored in time-series database) as input, and outputs an adaptive threshold through defined calculation rules to prevent false alarms caused by sudden power consumption spikes.

(3) Multi-endpoint synchronization implementation: Raspberry Pi receives threshold updates via persistent WebSocket connection, triggering OLED refresh. Mobile clients check for threshold changes via HTTP+WebPush long-polling, with local caching. Flask server provides threshold setting interface through REST API protocol

4.5 Display Module

OLED display is a self-emissive display technology used in the dormitory power management system to display real-time power data and warning status information. Its working principle is as follows:

1.OLED Display Principle

OLED display consists of millions of microscopic organic light-emitting diodes (pixels), each containing organic emissive materials. When voltage is applied, an electric field forms between the anode and cathode, causing electrons and holes to recombine in the organic layer and excite the organic material to emit light. Since each pixel can be independently controlled, no backlight is needed, enabling high contrast and low power consumption display.

2.SSH1106 Driver Control

In this project, we use an OLED display with SSH1106 driver chip, communicating with the main controller (e.g., Raspberry Pi) via I²C or SPI interface to achieve dynamic data refresh. The workflow is as follows:

- (1) Circuit connection: VCC to 3.3V, GND to ground, SCL to pin 5, SDA to pin 3.
- (2) Data input: The system obtains real-time power consumption data (such as remaining power, consumption rate) through the power monitoring module and converts it into displayable character or graphic format.
- (3) Command control: The main controller sends commands to SSH1106 to set display area (page management, 8 pages \times 128 columns total) and write corresponding pixel data (1=on, 0=off).
- (4) Dynamic refresh: SSH1106 continuously reads data from internal GRAM (graphics RAM) to drive OLED pixels, achieving real-time power information updates.OLED.

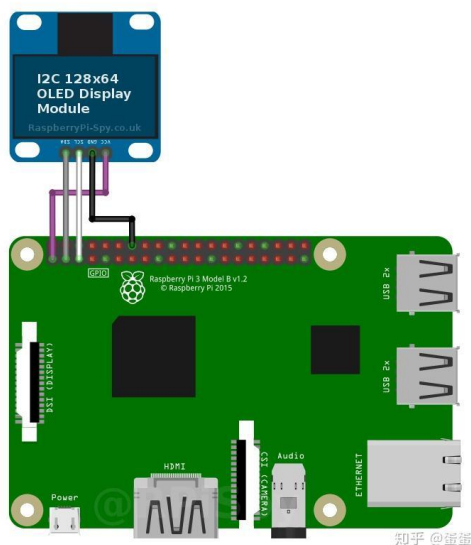


Figure 4-6: OLED circuit connection diagram

3. Functional Implementation

In this project, the OLED display clearly presents real-time power consumption data (e.g., "Current Power: XX kWh"). When the power level approaches the preset threshold, it automatically switches to inverse-color display or flashes warning symbols (e.g., "!!!!"). Additionally, it enters a low-power sleep mode (consuming only 0.1mA) during periods of inactivity and can be reactivated via human motion detection or button press.

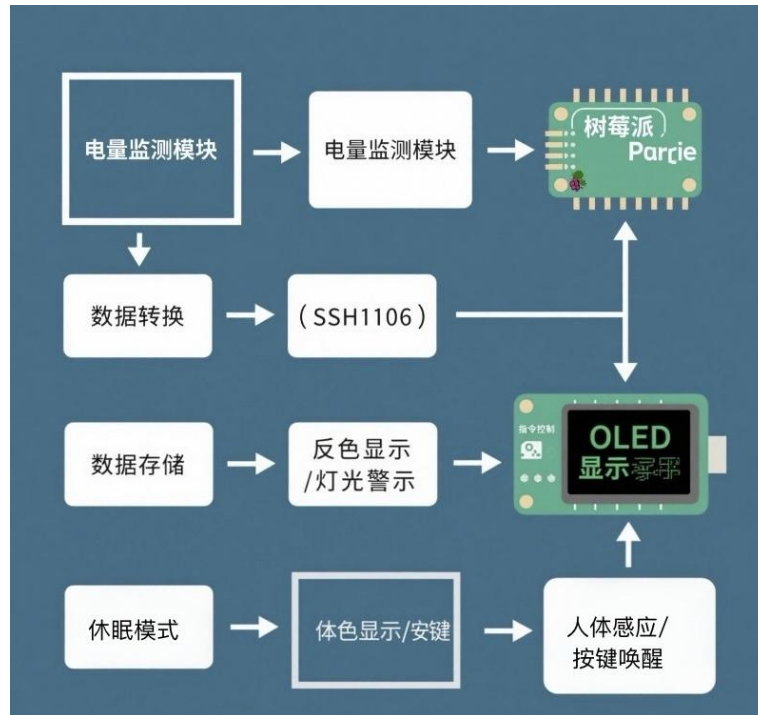


Figure 4-7: Display module architecture diagram

5. Test Cases and Execution Results

5.1 Dynamic Power Query and Real-Time Alert Display Function

First, open the campus mini-program to manually check the current dormitory power consumption, which is found to be 832.65kWh. Then run the system program. When starting the program for the first time, since obtaining power data takes some time, there is a certain delay in power acquisition. To avoid affecting user experience, "Loading..." is displayed on the screen to alleviate user anxiety.



Figure 5-1: Battery charging

After waiting for a period of time, the current remaining dormitory power is successfully obtained and matches our manual query result. It is worth noting that the right side of the figure below shows the front-end operation interface of this system.

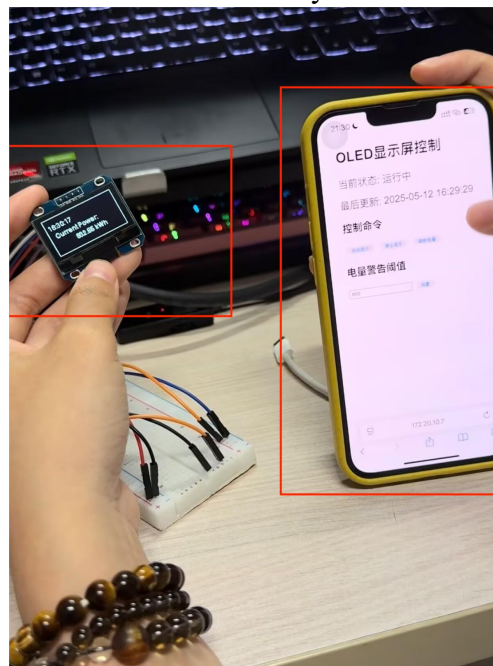


Figure 5-2: Battery level query successful

In the operation interface, we can view the current time and the last update time of power data. We can also use control commands to display power, stop display, and refresh power data. Additionally, we can dynamically set power warning thresholds. When the current remaining dormitory power falls below our set threshold, "!!!!!" will immediately appear in the upper right corner of the display to alert students to recharge in time.



Figure 5-6: Dynamic adjustment of battery warning threshold

5.2 Human Presence Detection and Robotic Arm Simulation Function

First, start the program normally. During initial operation, wait for the sensor to warm up for 30 seconds. Then wave your hand around the sensor to simulate someone passing by, and observe the servo rotating from bottom to top, successfully simulating the robotic arm opening the circuit breaker.

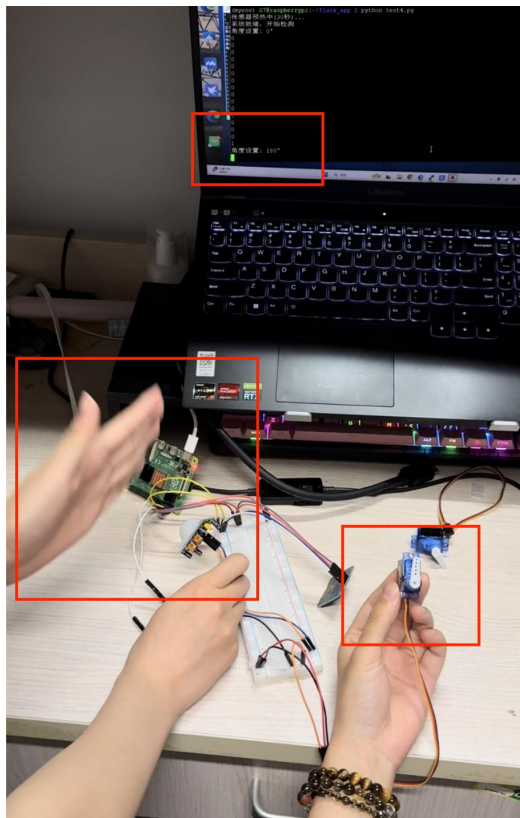


Figure 5-7: The steering gear rotates from bottom to top

Subsequently, lower your hand and observe the servo rotating from top to bottom, successfully simulating the robotic arm pulling down to close the circuit breaker.

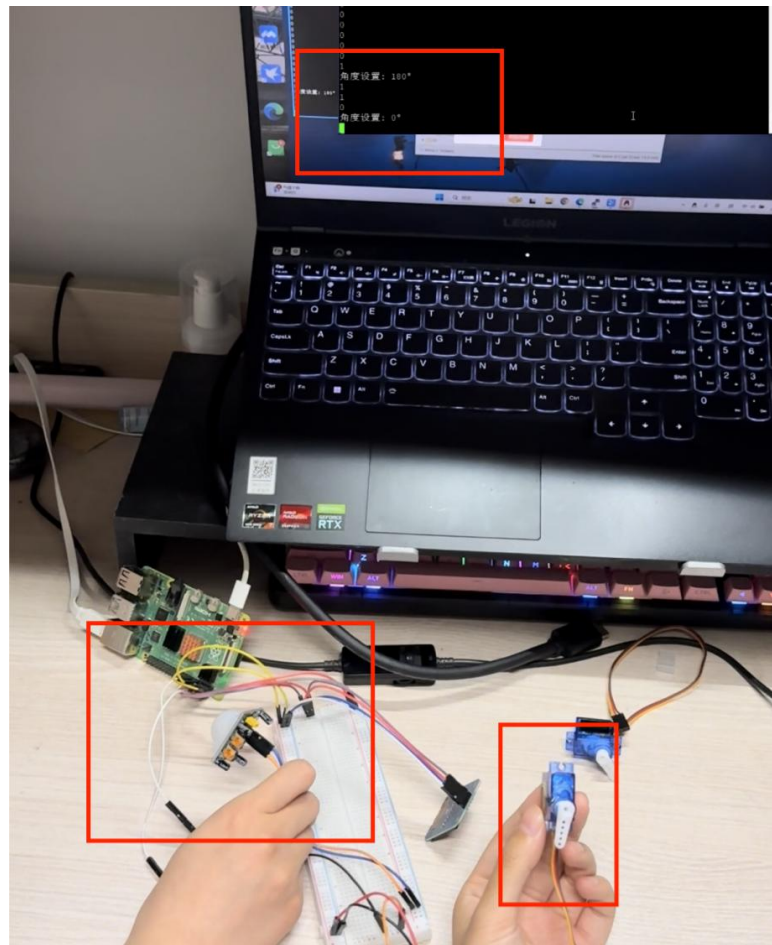


Figure 5-8: The steering gear rotates from top to bottom

5.3 Scheduled Operation Function

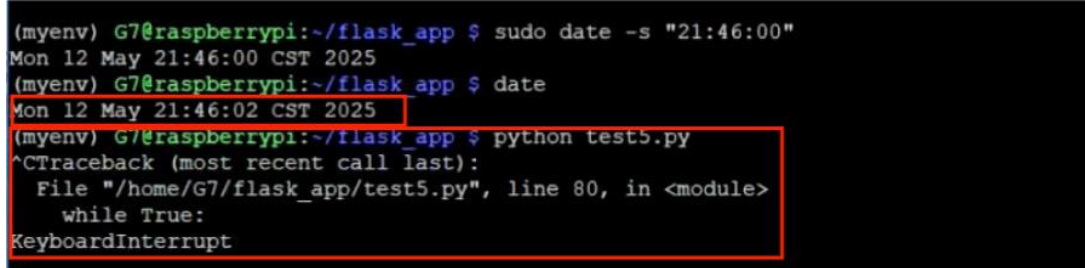
The system can periodically execute the human sensor detection and servo robotic arm simulation modules. Currently, we set the Raspberry Pi system time to 11:46 PM to simulate the period between 11:00 PM and 7:00 AM, and observed that the aforementioned system can automatically perform detection.

```
(myenv) G7@raspberrypi:~/flask_app $ sudo date -s "21:46:00"
Mon 12 May 21:46:00 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ date
Mon 12 May 21:46:02 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ python test5.py
^CTraceback (most recent call last):
  File "/home/G7/flask_app/test5.py", line 80, in <module>
    while True:
KeyboardInterrupt

(myenv) G7@raspberrypi:~/flask_app $ sudo date -s "23:46:00"
Mon 12 May 23:46:00 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ date
Mon 12 May 23:46:02 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ python test5.py
/home/G7/flask_app/test5.py:23: RuntimeWarning: This channel is already in use, co
tinuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(SERVO_PIN, GPIO.OUT)
传感器预热中(30秒)...
```

Figure 5-9: Simulate the working hours

We then set the Raspberry Pi system time to 10:46 PM to simulate non-operational hours, and found that the aforementioned system could not start. This confirms the successful implementation of the automatic scheduling functionality.

A terminal window on a Raspberry Pi. The prompt is (myenv) G7@raspberrypi:~/flask_app. The user enters 'sudo date -s "21:46:00"', and the output is 'Mon 12 May 21:46:00 CST 2025'. The user then enters 'date', and the output is 'Mon 12 May 21:46:02 CST 2025'. Next, the user enters 'python test5.py'. The output shows a KeyboardInterrupt error: '^C\nTraceback (most recent call last):\n File "/home/G7/flask_app/test5.py", line 80, in <module>\n while True:\nKeyboardInterrupt'.

```
(myenv) G7@raspberrypi:~/flask_app $ sudo date -s "21:46:00"
Mon 12 May 21:46:00 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ date
Mon 12 May 21:46:02 CST 2025
(myenv) G7@raspberrypi:~/flask_app $ python test5.py
^C
Traceback (most recent call last):
  File "/home/G7/flask_app/test5.py", line 80, in <module>
    while True:
KeyboardInterrupt
```

Figure 5-10: Simulate non-working hours

6. References

- [1] Dong Bin, Sun Dongliang, Cao Lichao, et al. Analysis of Self-Oscillation Caused by Servo Transmission Backlash and Its Suppression Strategies [J/OL]. Flight Control & Detection, 1-12 [2025-05-21]. <http://kns.cnki.net/kcms/detail/10.1567.tj.20250427.1437.004.html>.
- [2] Chang Zuoping. Automatic Switch Using Human Infrared Detection [J]. Household Appliances, 1996, (05): 9-10.
- [3] Zhang Yueyue, He Jiahao, Zhou Wenjing. Design of Automatic Human Detection Device Based on Multi-Sensor Fusion [J]. Technology Horizon, 2024, 14(24): 100-103.
- [4] Sheng Jing, Xu Chao, Zhou Tao. Technical Analysis of Python Web Scraping [J]. China Information World, 2024, (06): 210-212.
- [5] Zhao Pengxiang, Ma Zhichao, Liu Fei, et al. Research on Computer Network Security Protection Model Based on Embedded Development Technology [J]. Cybersecurity and Informatization, 2024, (12): 134-136.
- [6] Wang Yingfeng, Zhang Yanzhou, Gao Tao. Review of Energy-Efficient Scheduling for Avoiding Bus Conflicts in Multi-Core Embedded Systems [J]. Computer Applications Research, 2014, 31(04): 961-964+969.
- [7] Wang Lili, Song Xiaoqin. Research on Intelligent Power-Saving Control Based on Embedded Systems [J]. Science Technology and Engineering, 2016, 16(19): 255-258.
- [8] Li Ming, Chen Wei. IoT-Based Smart Electricity Management in Dormitories [J]. Modern Electronics Technique, 2025, 38(02): 182-184. DOI:10.20153/j.issn.2096-9759.2025.02.054.

7. Summary

7.1 What We Have Learned

In this project, our team delved into the complete development process of an Internet of Things (IoT) system, ranging from hardware selection and sensor data acquisition to the construction of cloud services and mobile interaction design. Through hands-on practice, we mastered embedded development technologies such as Raspberry Pi GPIO control, PWM servo drive, and the application of human body infrared sensors. We also became familiar with REST API design using the Flask framework, real-time communication via WebSocket, and database integration. The integration of these technical points has given us a more intuitive understanding of mechatronic systems, especially during the hardware-software co-debugging process, where we accumulated valuable troubleshooting experience.

In terms of system architecture, we attempted a distributed three-end communication design for the first time and understood the crucial role of the cloud server as a data hub. By implementing bidirectional communication between the Raspberry Pi and the Flask server (HTTP + WebSocket), we compared the differences in real-time performance and reliability among various protocols and learned to select appropriate communication solutions based on specific business scenarios. In particular, when dealing with issues such as network jitter and data synchronization conflicts, we significantly enhanced the system's robustness by designing state confirmation mechanisms and local caching strategies.

Most importantly, the project has cultivated our engineering thinking and team collaboration skills. From requirement analysis and module decomposition to code integration, we learned to manage collaborative development using version control tools and standardized front-end and back-end interactions through interface documentation. In the process of solving practical problems, such as sensor false triggering and insufficient servo torque, we not only improved our hands-on abilities but also gained a deeper understanding of the gap between theoretical knowledge and engineering practice. This interdisciplinary comprehensive training has significant guiding significance for future project development.

7.2 Problems Encountered and Solutions

During the implementation of the project, team members encountered several technical challenges. The following are some key issues and their solutions:

1. Raspberry Pi Configuration Issues

After upgrading the Raspberry Pi OS, the Python environment crashed, causing the GPIO library to fail. Additionally, the updated kernel version led to driver incompatibility, and the USB-to-serial chip required reinstallation of drivers.

To resolve these issues, we re-flashed the Raspberry Pi SD card. Subsequently, we used "pip freeze > requirements.txt" to lock the Python dependency versions. We also implemented

critical services as systemd services to enable automatic restarts in case of crashes and locked the specific system version in /etc/apt/source.list.

2. Inaccurate Human Body Detection by Infrared Sensors

When using sensors to detect human presence, inaccuracies often occurred. The main reasons included temperature interference, where the ambient temperature was close to human body temperature, resulting in reduced sensitivity of infrared radiation detection.

Our optimization solution involved paralleling a filtering capacitor ($0.1 \mu F$) at the signal output end to eliminate high-frequency jitter. The connection method was: OUT connected to the capacitor and then to GND. We also adjusted the sensitivity knob (SENS) clockwise to increase the detection range and turned the delay knob (TIME) counterclockwise to shorten the trigger hold time, avoiding continuous output. The jumper was set to the repeatable trigger mode (H).

3. Hardware Connection Issues

During the hardware connection phase, some components (such as LEDs and ultrasonic sensors) experienced poor contact. By inspecting the circuit design and ensuring all connections were secure, the problem was resolved. Additionally, we used more robust DuPont wires and connection methods to prevent system instability caused by hardware failures.

7.3 Team Improvement

Throughout the project, team members not only significantly enhanced their technical skills but also received valuable training in project management, communication, and problem-solving. Through collaboration, everyone learned how to divide tasks and cooperate effectively, while also deepening their understanding and application of technology. Regular discussions and testing helped us avoid many potential issues and allowed for timely adjustments and optimizations at different stages of the project.

7.4 Areas for Improvement

Although we have successfully implemented all system functions, there are still some areas that need further enhancement compared to practical application requirements:

1. Cloud Management Platform

Currently, the system allows mobile access through the Flask server, but it requires the mobile device and Raspberry Pi to be on the same router, preventing remote viewing. Therefore, we can consider integrating a cloud management platform in the future. This would not only enable remote querying of electricity data and control of dormitory switches but also support centralized viewing and management of multiple dormitories, thereby improving management efficiency.

2. Mechanical Arm Issues

The system's simulation of a mechanical arm using S90 servos is relatively primitive and may lead to hardware wear or insufficient force to operate the switch over time. We can consider using electronic circuit breakers in the future to enhance the system's stability and

safety.

3. System Response Time

Although multi-process concurrent control has reduced the threshold update to screen display time to less than 500ms, the time for electricity data acquisition has not yet reached the millisecond level. We need to continue optimizing the electricity data crawling technology, such as using session persistence and connection pooling to optimize the network layer and incremental crawling to improve the technical architecture.

4. Data Security

When extracting dormitory electricity data, security issues are undoubtedly involved. In the future, we should add firewall settings to the system and implement password authentication for web login to ensure the confidentiality of student data.

7.5 Future Learning Plan

Moving forward, team members plan to further deepen their learning in the following areas:

1. IoT and Smart Hardware

In the field of IoT, the team will learn more about sensors, embedded development boards (such as Arduino and ESP32), and cloud computing platforms to enhance the project's remote monitoring capabilities and system intelligence.

2. System Integration and Optimization

The team will also focus on overall system optimization to reduce latency, improve accuracy, and further enhance user experience. Additionally, considering the diversity of real-world application scenarios, the team will explore integrating more APIs to expand system functionality (such as adding voice reminders and automatic electricity bill payment features) and improve the system's intelligence and adaptability.

8. Project Division of Labor

1. Li Qianqian

- **Raspberry Pi and Network Configuration:** Responsible for initially connecting the Raspberry Pi to the local computer and creating a virtual local area network to ensure all team members can operate the Raspberry Pi normally. During the middle stage, solve all network issues to ensure normal network configuration and accessible interfaces.
- **Electricity Data Crawling:** Write Python code for electricity data crawling, obtain the interface of the school's electricity bill inquiry mini-program, and periodically access it. Use xdotool to simulate clicks to crawl the remaining electricity in the dormitory and return this data to the next module.
- **File Monitoring and Dynamic Threshold Warning Mechanism Coding:** Integrate the OLED display module with the electricity data crawling module and structurally combine it with the Flask control system. Implement a dynamic threshold warning mechanism through a distributed three-tier structure to achieve efficient multi-process collaboration.
- **Server Deployment:** Integrate all functional modules and deploy the main program to the Flask server to ensure that the mobile end can operate the system interface normally and return the operation results to the Raspberry Pi control end.
- **System Operation Interface Design:** Design the front-end operation interface of the system to ensure users can control the Raspberry Pi and its various module functions through the mobile end.
- **Drafting Software Documentation and Project Report:** Responsible for the initial drafting of software documentation and project reports.

2. Han Jing

- **Servo Simulated Mechanical Arm Behavior Module Development:** Design and implement the control code for the SG90 servo to ensure the Raspberry Pi can control the SG90 servo through GPIO pins to simulate a mechanical arm for switching the power gate.
- **Network Configuration:** Assist with network configuration issues between the Raspberry Pi, PC, and mobile end.
- **Integration of Sensors and Servo:** Integrate the human body infrared sensor with the SG90 servo module to enable the system to detect human presence in real-time and control the servo to rotate up and down to simulate the mechanical arm's operation of switching the power gate.
- **Timed Detection Function Development:** Set and modify the Raspberry Pi's system time and write code for the timed module to enable the system to automatically detect the presence of people in the bathroom at night and control the mechanical arm to switch the power gate.
- **Literature Collection and Data Search:** Collect various literature in the early stages of the project to provide strategic planning for the project.
- **Optimize Project Report:** Review and optimize the project report in the later stages.

3. Miao Chunxu

- **OLED Display Module Development and Testing:** Design and implement the OLED display module, responsible for writing code and maintaining and testing it later to ensure the Raspberry Pi can display the remaining electricity in the dormitory in real-time through GPIO pins and I2C.
- **Circuit Connection:** Responsible for purchasing hardware devices, debugging hardware tools, and connecting various circuits to ensure the system runs normally.
- **Real-Time Electricity Display Function Development:** Assist in integrating the OLED display module with the electricity data crawling module and implement a dynamic threshold warning mechanism through a distributed three-tier structure to achieve efficient multi-process collaboration.
- **Market Research and Competitive Analysis:** Collect data on similar products in the market and conduct competitive analysis.
- **Identify Project Shortcomings and Future Optimization Areas:** Review and debug the project in the later stages and identify areas for future optimization through competitive analysis.
- **Draft and Optimize Project Reports and Software Documentation:** Responsible for writing and revising project reports, reviewing and optimizing software documentation to ensure they meet standards and are highly readable.

4. Tan Mengtong

- **Human Body Sensor Recognition Module:** Design and implement the human body sensor recognition module, responsible for writing code and maintaining and testing it later to ensure the Raspberry Pi can detect human presence in real-time through GPIO pins and return the results to the next module.
- **Circuit Connection:** Debug hardware tools and connect various circuits to ensure the system runs normally.
- **Integration of Sensors and Servo:** Assist in integrating the human body infrared sensor with the SG90 servo module to enable the system to detect human presence in real-time and control the servo to rotate up and down to simulate the mechanical arm's operation of switching the power gate.
- **Flowchart Drawing:** Draw various flowcharts for the project to help team members better understand the system's working principles.
- **Optimize Project Report:** Review and optimize the project report in the later stages.
- **Literature Review:** Collect various literature in the early stages of the project to provide strategic planning for the project.