Research article
DOI: https://doi.org/10.18721/JCSTCS.18406
UDC 621.397

DESIGN OF IOT DEVICE USING BEAM-SPLITTING PRISM DISPLAY

Xu Luolan ✉

Xi'an Jiaotong University, Xi'an, China

✉ xuluolan123@gmail.com

Abstract. This project is based on an open-source hardware. Its functions have been redesigned and improved, with enhancements made to the hardware programming and power supply circuits. System-level secondary development has been carried out to implement an embedded system interface and a weather clock application. During development, modifying device functionality required close coordination between software and hardware and optimization of the enclosure structure. As resource demands increases, adjustments to the hardware power supply and software optimization become necessary to ensure reliable operation. Internet of things (IoT) device development necessitates a holistic approach. By working backward from the target specifications, hardware, software and enclosure design must be considered together. The results indicate that optimized power supply, low-coupling software operation and a thermally efficient enclosure significantly enhance the long-term stability and low-power operation of IoT devices.

Keywords: embedded systems development, internet of things (IoT), devices, beam-splitting prism display, low-power optimization, graphical user interface (GUI)

Acknowledgements: I would like to thank Peng Zhihui, the author of the open-source project, the users of the project's forum, my university and its teaching staff for their assistance in the implementation of this research.

Citation: Xu Luolan. Design of IoT device using beam-splitting prism display. Computing, Telecommunications and Control, 2025, Vol. 18, No. 4, Pp. 67–75. DOI: 10.18721/JCSTCS.18406

Научная статья
DOI: https://doi.org/10.18721/JCSTCS.18406
УДК 621.397

РАЗРАБОТКА УСТРОЙСТВА IOT С ИСПОЛЬЗОВАНИЕМ ДИСПЛЕЯ НА ОСНОВЕ ЛУЧЕПРЕЛОМЛЯЮЩЕЙ ПРИЗМЫ

Л. Сюй ✉

Сианьский транспортный университет, Сиань, Китай

✉ xuluolan123@gmail.com

Аннотация. Настоящий проект основан на аппаратном обеспечении с открытым исходным кодом. Его функции были переработаны и улучшены, включая усовершенствование схем аппаратного программирования и питания. Была проведена дополнительная разработка на системном уровне для реализации встроенного системного интерфейса и программы для метеостанции с часами. В ходе разработки модификация функциональности устройства потребовала тесной координации между программным и аппаратным обеспечением, а также оптимизации конструкции корпуса. При увеличении потребления ресурсов для обеспечения надежной работы потребовалась настройка аппаратного обеспечения и оптимизация программного обеспечения. Разработка устройств Интернета вещей (IoT) требует комплексного подхода. Исходя из целевых технических характеристик, необходимо рассматривать в комплексе аппаратное, программное обеспечение и дизайн корпуса. Результаты показывают, что оптимизированный источник питания, программное обеспечение с низким энергопотреблением и термоэффективность корпуса значительно увеличивают срок службы и улучшают стабильность работы IoT-устройств.

Ключевые слова: разработка встраиваемых систем, устройства, интернет вещей (IoT), дисплей на основе лучепреломляемой призмы, оптимизация энергопотребления, графический интерфейс пользователя (GUI)

Благодарности: Я хотел бы поблагодарить Пэн Чжихуэя, автора проекта с открытым исходным кодом, пользователей форума проекта, мой университет и его преподавательский состав за помощь в реализации данного исследования.

Для цитирования: Xu Luolan. Design of IoT device using beam-splitting prism display // Computing, Telecommunications and Control. 2025. Т. 18, № 4. С. 67–75. DOI: 10.18721/JCSTCS.18406

Introduction

With the rapid development of information technology, the proliferation of Internet of things (IoT) devices has permeated various aspects of personal life, such as smart home and smart in-car systems. Therefore, the rapid development of versatile IoT interactive terminal devices capable of connecting to various gateways holds significant research value and importance [1].

This paper explores the secondary development of an open-source beam-splitting prism display device, enabling beginners to learn and master hardware design and optimization, as well as secondary software development and comprehensive understanding of embedded systems [2].

Progress in electronics often comes from standing on the shoulders of giants. Open-source projects represent the cumulative effort of numerous developers gradually refining a design. This study focuses on creating an IoT device based on ESP32 hardware circuits and beam-splitting prism display, involving circuit optimization, program modification, and functional enhancements such as temperature control and weather information display. By creating an actual application, the device can later be extended into a multifunctional integrated system – hence the project name AIO (All-In-One).

Research design

The ESP32-PICO-D4 was selected as the core controller for this research. It features a dual-core 32-bit processing core, RTC and low-power management module [3], and provides complete Wi-Fi and Bluetooth functions, which greatly enhances the device's practicality.

For the display, given the current trend toward transparent frameless screens, a balance between technology and cost-effectiveness was sought. The method of prism refraction display can be used to present various functions in a "transparent" manner at low cost. This project's name AIO emphasizes that the hardware platform serves as a foundation upon which a wide range of functions – such as the weather and time display in this implementation – can be added through secondary software development.

Hardware design

Based on the open-source hardware, this project focuses on implementing weather and time display with network NTP synchronization. The original hardware design supported only basic image display functions, resulting in relatively low power consumption. However, the long-term simultaneous operation of network time synchronization, screen display, accelerometer, RGB lights and other components will greatly increase the overall circuit power consumption, requiring modifications to the hardware power supply [4].

The updated hardware design is as follows.

A two-layer printed circuit board (PCB) design was adopted. The upper PCB is designed for screen display and optimized power supply. According to the size of the prism, a 1.3-inch IPS color display with a 240×240 resolution (square format) was selected, ensuring the display area aligns precisely with the prism's refracting region. The lower PCB integrates main controller, accelerometer, RGB lights and TransFlash card slot. The two circuit boards are connected by flexible flat cable (FFC) connectors, which provide power and signal communication. Fig. 1 shows the front view of the lower main control board of the device.

For the upper voltage-stabilizing display module of the device, the original circuit's voltage-stabilizing chip failed to provide stable power supply, and it was replaced with the ME6211 linear voltage regulator to supply power to the entire device. Fig. 2 shows the front-side layout of the voltage-regulator and display board.

Since there are no physical buttons, all control and interaction rely on the MPU6050 accelerometer, which is placed in the middle of the lower main control board circuit to ensure accurate parameters during program initialization. The ESP32 and the CP2102 chip (used for programming)

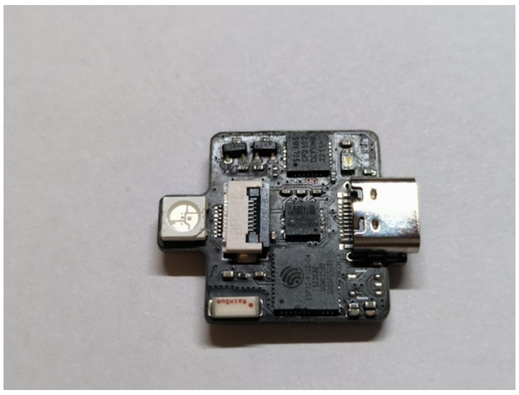

Fig. 1. Lower main control board

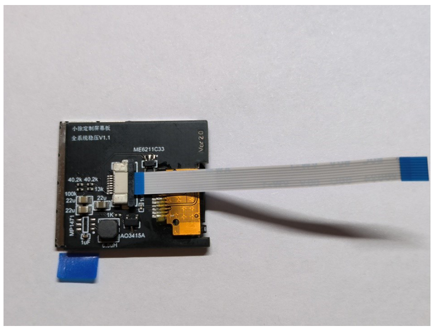

Fig. 2. Front-side layout of the voltage-regulator and display board

are placed on the upper and lower sides of the accelerometer, respectively, while the Type-C port and RGB LEDs are placed on the left and right sides. The back of the main control board is designed with a TF card slot for storing images and videos in later stage, as well as files for development and calling. This helps to save the memory space of the MCU itself.

The original project used the LP2992 linear voltage regulator, which supports a maximum output current of 250 mA. However, the screen alone requires approximately 100 mA when illuminated. Moreover, subsequent maintenance programs occupy a large amount of processing capacity, leading to excessive power consumption that can cause severe heating of the low-dropout (LDO) regulator and even program crashes. To solve this problem, this project replaced the ME6211C33 linear voltage regulator chip in the screen's power supply board, increasing the load current capacity to 500 mA and thereby resolving the issue of overheating of the original IC. The optimized parameters of the updated circuit are shown in Table 1.

Table 1

Comparison of updated device parameters

	Original	**Modified**
Performance parameters	LP2992 3V3	SGM662K-3.3
Maximum output current	250 mA	**500 mA**
Typical dropout voltage	115 mV @ 100mA 200 mV @ 150mA	**130 mV @ 300 mA** **200 mV @ 500 mA**
Quiescent current	Typical 170 μA	**Typical 30 μA**
Ripple rejection ratio	75 dB @ 1kHz 45 dB @ 100kHz	75 dB @ 1 kHz 40 dB @ 10 kHz
Protection functions	**Overcurrent, overtemperature, reverse current protection, enable control**	Overcurrent, overtemperature, reverse current protection
Input voltage range	**2.5V ~ 16V**	2.2V ~ 5.5V
Operating junction temperature range	**–40°C ~ +125°C**	–40°C ~ +85°C
Circuit ripple	22 mV	**18 mV**
Cost	Relatively high	Very low

It can be seen that the updated power supply IC provides optimization in terms of low power consumption, high-current drive capability and cost-effectiveness. Additionally, actual tests show that the power ripple across the overall power supply has been reduced.

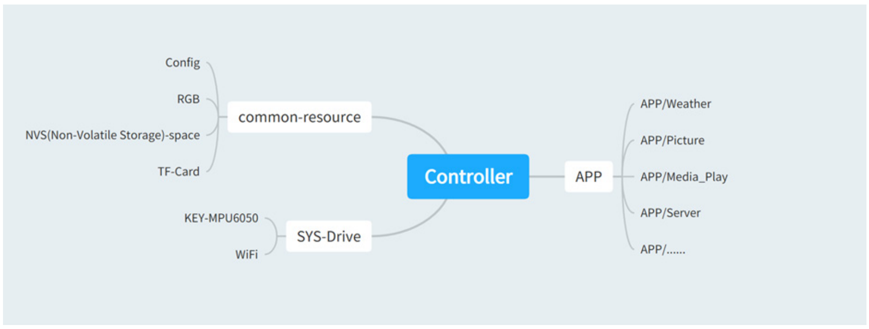

Fig. 3. System design mind map

Software framework

Based on the hardware design, the software must account for the actual computing power of the MCU. Therefore, the light and versatile graphics library (LVGL) was selected to handle the control display interface, which also helped shorten the software development cycle. The logical design of the system framework can be understood from the overall system design mind map shown in Fig. 3.

It can be seen that all functions are implemented through the main control class (system graphical interface). Since the hardware uses an accelerometer for control, the sensor's output is processed as the key value to realize human-machine interaction logic, which is the core control method of the device – operation via the gyroscope sensor. At the software level, the key value can be used as a controller to select the running app code module, thereby realizing different functions [5]. Once inside the app, the key value is also used for control or selection. In this way, the integrated sensor fully replaces various external devices such as computer mouse or keyboard, making the device very compact and fully functional – consistent with the design philosophy of small, portable IoT devices.

Each app (structured as a module) can access public resources as long as it meets the interface specifications of the overall framework. For example, access to the TF card, RGB lights, global configuration files, network settings and small system parameters can all be modified and controlled. This low-coupling program design enables rapid customization of software according to existing hardware resources during app development.

It should be noted that, due to the physical characteristics of the light-splitting prism, during the system development process, the image displayed on the screen must be mirrored to display the correct content that appears on the front side of the prism.

Implementation of weather system app function

On the 1.3-inch color display, in addition to text data, weather icons can be added to enhance the visual effect [6]. This requires the use of the display font library and the integration of the LVGL [7]. Since this project only uses weather and region-related content, other elements of the library can be removed to save processor storage space.

For example, Chinese Amap Open Platform provides real-time updates of temperature, humidity, wind strength and air quality for specified cities, as well as weather forecast information for the next week. The app ID, app secret , city name, and network synchronization interval information are recorded in the ESP32 to avoid repeated settings after power loss or restart [8].

However, since weather conditions seldom change over short periods of time, displaying this information is unnecessary. Moreover, this may lead to screen burn-in. Therefore, this project incorporates dynamic clock display. On one hand, it can update data in real time to prevent screen burn-in; on the other hand, it enhances the visual appeal of the weather app.

The weather app consists of the following modules, coordinated via functions or messaging mechanisms:

1. Initialization Phase

   a. Read or write weather-related configurations (e.g., city name, update interval).

   b. Prepare basic styles such as fonts and color schemes for the interface.

   c. Initialize runtime data for storing weather and time information.

2. Main Loop/Event Processing

   a. Continuously monitor user input (e.g., return, left/right switching) and system messages (e.g., Wi-Fi connection, parameter settings) in a loop or timed callbacks.

   b. Determine the next action based on the current page type (weather or curve page): update weather data or time, or simply display existing data.

3. Data Acquisition and Parsing

   a. When the network is available, access the corresponding weather and time API; parse JSON responses or timestamps.

   b. When the network is unavailable, use local millisecond counters for time calculation until connection is restored.

   c. Store the obtained or calculated results in the runtime data structure.

4. Interface Display and Refresh

   a. Update graphical interface when new data is available.

   b. On the weather page: display current temperature, humidity, wind strength, city name, etc., along with animations.

   c. On the curve page: draw weekly maximum/minimum temperature curve.

   d. When the network is available, update the clock display (hours, minutes, seconds, date).

5. Exit/Cleanup

   a. When the user or system requests to exit, destroy interface objects and styles, close background tasks, release memory and ensure that no system resources are retained.

The flowchart of the weather program design is shown in Fig. 4. It should be noted that the graphical interface of the weather app is developed based on LVGL, including interface and style, multi-page management, weather display, forecast curve, clock and date display, animation effects, etc. This project draws inspiration from the clock interface template made by Misaka. Image design is customizable according to personal preferences and will not be described in detail in this paper.

Shell design

Since the designed IoT device needs to run continuously and the ESP32 periodically performs network time calibration through Wi-Fi, heat generated by the power supply and screen display will inevitably accumulate [3]. If heat dissipation cannot be guaranteed, the operation stability of the IoT device will be reduced [9]. Therefore, the shell connecting the main control PCB, screen PCB and upper light-splitting prism will assume the main heat dissipation function. In the early stage, 3D printing can be used to determine the rationality of the hardware structure distribution. In the later stage, the designed structure can be made of aluminum alloy, and the shell model parameters can be delivered to a factory capable of CNC machining [10]. Thanks to technological advancements and the convenience of online shopping, the cost of small-batch design and processing has been greatly reduced. It should be noted that due to the adoption of a metal heat dissipation shell design, the PCB should use the via tenting process, and during assembly, the metal parts on the circuit board must be properly insulated for the circuit.

Results

After powering on the assembled device, it first boots into the system's main interface. From there, the weather app can be selected and entered through tilt-based navigation. After entering the interface,

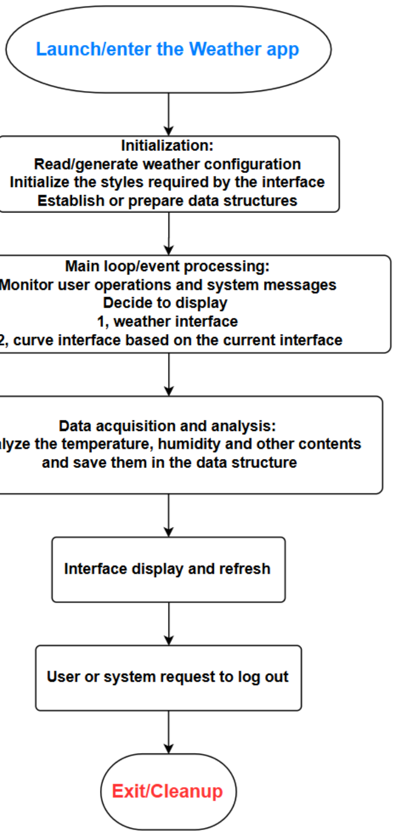
Launch/enter the Weather app
Initialization:
Read/generate weather configuration
Initialize the styles required by the interface
Establish or prepare data structures
Main loop/event processing:
Monitor user operations and system messages
Decide to display
1, weather interface
2, curve interface based on the current interface
Data acquisition and analysis:
Analyze the temperature, humidity and other contents
and save them in the data structure
Interface display and refresh
User or system request to log out
Exit/Cleanup

Fig. 4. Weather app design flowchart

the program initially displays the default time and weather screen, and the time is updated according to the RTC built into the ESP32, which ensures normal app operation. Shortly after, the device experiences a brief lag, while the program is calibrating the weather and time data over the network. Subsequently, the interface will display the synchronized information, indicating that the overall design functions are operational. After 24 hours of continuous operation, as shown in Fig. 5, the displayed time remains accurate to the second due to regular calibration, with no observed desynchronization. This also demonstrates stable performance of the device's network and weather display functions.

By touching the metal shell part, it is evident that most of the heat generated by the device is transferred to the shell, and the temperature is stably controlled at a level slightly above ambient level, indicating that the effective auxiliary heat dissipation is provided by the shell.

Finally, all content is presented in the transparent glass through the refraction and reflection of the prism.

Conclusion

Through the design and implementation of this prism-based display IoT project, the intended functions have been realized via integrated hardware and software design, shell design, and novel prism display design. A deeper understanding of the comprehensive optimization of device stability has been obtained. This project-based learning approach has proven to be a highly effective methodology.

During hardware design, it was found that after software updates, the screen failed to display properly due to insufficient current; the power supply was subsequently modified. During software design, app development was carried out using a low-coupling approach to reduce redundant code. During shell design, overheating was found to cause system crashes, and the shell material was improved to

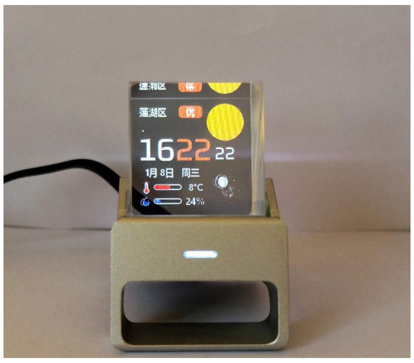

Fig. 5. Equipment operation

enhance overall thermal conductivity. These modifications were identified and implemented during the secondary development of the original device.

Compared with the initial hardware design, the combination of optimized voltage-stabilizing circuit, metal shell and main control board heat dissipation realizes more stable voltage control, thereby improving the stability of the module during long-term use and reducing circuit ripple noise. At the software level, by setting public reusable interfaces, subsequent development on this hardware platform can incorporate customized apps within the overall framework. The IoT module can adapt to different usage scenarios by software upgrades, significantly reducing the app development cycle of the underlying operating system.

In the future, the number of IoT devices is expected to show explosive growth and will appear in all aspects of people's lives in various forms [8]. In terms of interaction logic, design will continue to engage human visual, motion, auditory, and tactile perception [5]. Presenting the interface in a more elegant manner remains a persistent goal for engineers. Based on the design of the minimum core module of the IoT, this project presents an interactive form different from traditional screen display. It is hoped that through this way, open-source IoT devices can show a more diversified development.

REFERENCES

1. **Jocknoi L., Kucharoen P.** ESP32Exten: Designing and developing an ESP32 microcontroller expansion for IoT applications with motor propulsion and AI image processing. *2024 8th International Conference on Information Technology* (*InCIT*), 2024, Pp. 278–283. DOI: 10.1109/InCIT63192.2024.10810578

2. **Wang Z., Tu K., Lv G., Feng Q.** Depth enhanced holographic super multi-view display based on multiple image recording planes. *IEEE Journal of Selected Topics in Quantum Electronics*, 2024, Vol. 30, No. 2, Art no. 6000506. DOI: 10.1109/JSTQE.2024.3364581

3. **Farid N.A.M., Razak A.H.A., Halim A.K., Idros M.F.M.** Design of CMOS RF Doherty power amplifier in low-power 5G wireless networks for IoT application. *2023 IEEE 11th Conference on Systems, Process & Control* (*ICSPC*), 2023, Pp. 310–314. DOI: 10.1109/ICSPC59664.2023.10420155

4. **Saxena A., Haripriya D., Madan P., Srivastava A.P., Shalini N., Kumar A.** Design and optimization of low-power VLSI circuits for IoT devices. *2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering* (*UPCON*), 2023, Pp. 1267–1273. DOI: 10.1109/UPCON59197.2023.10434775

5. **Nwadiugwu W.P., Kim D.-S.** Energy-efficient sensors in data centers for Industrial Internet of Things (IIoT). *2018 3rd International Conference on Internet of Things: Smart Innovation and Usages* (*IoT-SIU*), 2018, Pp. 1–6. DOI: 10.1109/IoT-SIU.2018.8519871

6. **Devi A., Arivunambi A., Suvetha S., Sasikala S., Dharanyadevi P., Senthilnayaki B.** IoT based satellite balloon system for live-weather forecast. *2024 3rd International Conference on Smart Technologies and Systems for Next Generation Computing* (*ICSTSN*), 2024, Pp. 1–5. DOI: 10.1109/ICSTSN61422.2024.10671106

7. **Zaharia S., Rebedea T., Trausan-Matu S.** Source code vulnerabilities detection using loosely coupled data and control flows. *2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (*SYNASC*), 2019, Pp. 43–46. DOI: 10.1109/SYNASC49474.2019.00016

8. **Cheng Z.** Research on Internet of Things human-computer interaction system based on computer Artificial Intelligence technology. *2024 IEEE 2nd International Conference on Control, Electronics and Computer Technology* (*ICCECT*), 2024, Pp. 1135–1139. DOI: 10.1109/ICCECT60629.2024.10545728

9. **Wijittemee W., Plangklang B.** An efficient thermal management using passive cooling techniques. *2024 International Conference on Power, Energy and Innovations* (*ICPEI*), 2024, Pp. 73–77. DOI: 10.1109/ICPEI61831.2024.10748605

10. **Boora A., Verma D., Bijender, Soni A.** Time reduction analysis based on infill pattern on FDM 3D printed PLA material. *2024 International Conference on Intelligent & Innovative Practices in Engineering & Management* (*IIPEM*), 2024, Pp. 1–4. DOI: 10.1109/IIPEM62726.2024.10925750

INFORMATION ABOUT AUTHOR / СВЕДЕНИЯ ОБ АВТОРЕ

Xu Luolan
Сюй Лолань
E-mail: xuluolan123@gmail.com

Submitted: 10.01.2025; Approved: 13.11.2025; Accepted: 09.12.2025.
Поступила: 10.01.2025; Одобрена: 13.11.2025; Принята: 09.12.2025.