HAL open science

Design and Development of Greenhouse Energy Management Platform Based on STM32

Junlin Sun, Xin Zhang, Cheng Zeng, Wengang Zheng, Lipeng Guo, Yali Du

To cite this version:

Junlin Sun, Xin Zhang, Cheng Zeng, Wengang Zheng, Lipeng Guo, et al.. Design and Development of Greenhouse Energy Management Platform Based on STM32. 10th International Conference on Computer and Computing Technologies in Agriculture (CCTA), Oct 2016, Dongying, China. pp.160-172, 10.1007/978-3-030-06155-5_16 . hal-02179982

HAL Id: hal-02179982

https://inria.hal.science/hal-02179982

Submitted on 12 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Design and development of greenhouse energy management platform based on STM32

Junlin Sun[2], Xin Zhang [1](✉), Cheng Zeng [2], Wengang Zheng[1], Lipeng Guo[3], Yali Du [2]

[1] Beijing Research Center for Intelligent Agricultural Equipment, Beijing, China
{zhangx,zhengwg}@nercita.org.cn
[2] Electronic and Information Engineering, Hebei University of Technology, Tianjin China
{379473744, 1135240956}@qq.com,zch@hebut.edu.cn
[3]Shijiazhuang Academy of Agriculture and Forestry Sciences, Shijiazhuang China
576837620@qq.com

Abstract. Greenhouse energy management system can make some systemic administration such as measure and analysis for water flow, electric quantity and quality, heat energy, gas flow and some other energy parameters in greenhouse for achieving a certain energy-saving effect. The introduction of the technology is mostly based on PC platform. The embedded greenhouse energy management platform based on STM32 was designed. This paper gave the system hardware and software architecture including the selection of hardware, transplantation of μCOS-III, transplantation of embedded database, design of human-machine interface and the main process of software. Finally, the platform is verified by STM32-V5 development board of Arm fly. The results show that the platform can meet the basic needs of energy management in greenhouse, and it has the advantages of low cost, strong expansibility and so on, it is feasible and effective to apply it to the greenhouse industry in China.

Keywords: Greenhouse · Energy · Management · Embedded system · Energy conservation · μCOS-III

1 Introduction

Energy management system is developed in the 1990s, a system of energy-saving technology, which utilize the processing control theory, network technology, optimization theory and the technology of database to conduct a comprehensive monitoring for energy systems, and to provide the basis for energy scheduling and allocation for achieving the purpose of efficient use of energy and energy conservation[1]. At present, energy management system has been widely used in iron and steel enterprises, office building, home and some other fields[2-8], through the monitoring, measurement, evaluation and analysis of the energy consumption, the optimization of the energy saving control strategy has been worked out, which has achieved remarkable results[9].

In order to solve the problem of greenhouse energy consumption, some horticultural developed countries have carried out research on energy management system in the greenhouse and achieved certain results. The greenhouse energy management system of ARGUS company in Canada, through the monitoring of

greenhouse environment and energy consuming equipment, can detect abnormal energy consuming equipment and alarm, and make a optimization strategy suitable for the specific conditions according to the analysis of the data, comprehensive energy saving up to 30% to 40%; The system designed by PRIVA company in Netherlands, using advanced astronomical time control, delay technology, integral blind control technology and control strategy, give full consideration to the relationship between processing control and the whole system up to save 50% of energy [10].

The majority of domestic greenhouse has no supporting energy management system, although some large greenhouse also uses the corresponding energy management technology, but the technology is mainly introduction in foreign countries, and due to the different climate and geological conditions as well as the specific circumstances, the introduction of energy-saving technology is not entirely suitable for China's national conditions [11], low efficiency, high energy consumption and the problem has not been resolved. Thus, the situation of greenhouse energy consumption in China is grim, specific problems are as follows:

(1) diverse types of greenhouse, backward facilities, lack of metering equipment, difficult to deploy the application of energy management system;

(2) bias on the structure of greenhouse, lack of suitable greenhouse energy management system, resulting in the low energy utilization rate, energy cost is increasing year by year;

(3) the introduction of foreign greenhouse energy management systems are mostly based on PC platform development, system operation is complex, does not conform to embedded system development needs, poor scalability, narrow application.

In view of the above problems, this paper designs the embedded greenhouse energy management platform based on STM32 for energy management of greenhouse in China. On the basis of analyzing the basic function requirements of the energy management, the software and hardware design framework of the whole platform is presented. The platform uses embedded operating system μCOS- III to realize multi task real-time scheduling, the SQLite database is transplanted to store the energy data, and the human-machine interaction interface is designed based on μC/GUI. After verification, the platform can meet the basic needs of energy management in the greenhouse, and has the advantages of low cost, strong expansion, suitable for China's national conditions and so on.

2 System Architecture

2.1 Functional requirements analysis

The system used intelligent electric meter, flow meter and some other energy metering equipment to gather the energy consuming data of greenhouse internal water, electricity, heat, etc. Then, the data was transmitted to the embedded terminal and directly reflected to the user in form of graphs and reports, and according to the analysis of these data, the system could provide energy saving scheduling decision support. Therefore, the overall function of the greenhouse energy management system can be divided into "energy measurement", "energy consumption analysis", "decision support" and "system settings", the specific function framework is as shown in Fig.1.

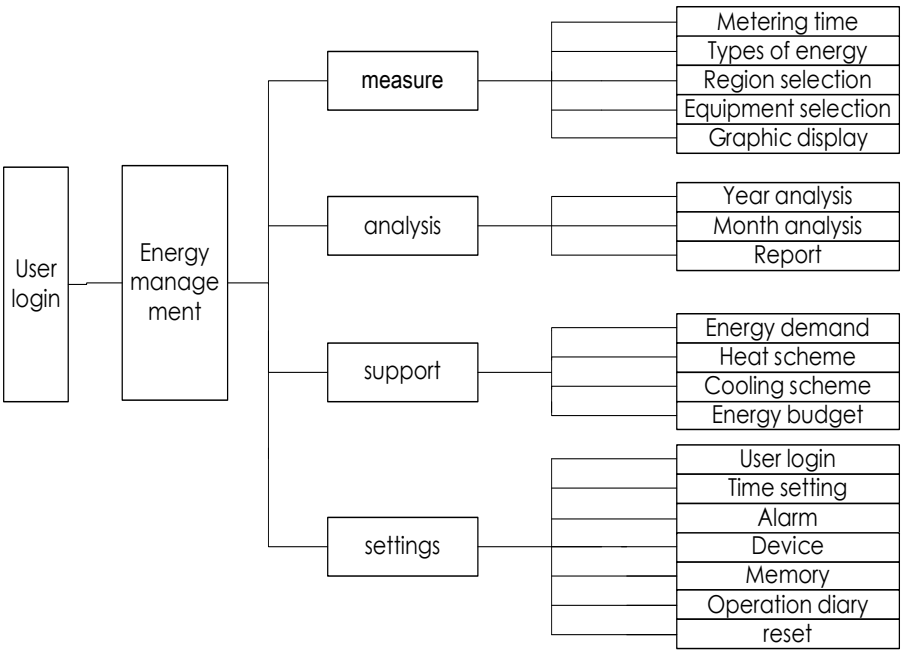

Fig.1. Basic function framework of greenhouse energy management system

In the module of “energy measurement”, system completed the measurement of energy consumption including water, electricity, heat, gas and so on. The user could select the time of measurement, the type of energy, the measuring area, the measuring equipment and the type of graphic display; The analysis module could provide energy consumption analysis compared with the same period last year and provide consumption analysis for several months, it could also give a report form according to the needs of users; The support function was an extension module, which could make different optimization schemes according to different conditions of the greenhouse. General decision support mainly included whether there was heating or cooling demand, energy demand budget and recommended heating/cooling program; In the "Settings" module, there were some common setting options such as “user login”, “time setting”, “device options”, “alarm options” and so on.

2.2 Software and hardware architecture

The basic work flow of energy management system was as follows: The system collected the consumption information of water, electricity, heat, gas and other energy in greenhouse by intelligent electric meter, flow meter and other data acquisition equipment, then, the data was transmitted to the central processing unit and stored to the data storage module, and finally through the human-machine interaction module to complete the interaction with the user. Therefore, in order to meet the basic functional requirements of the energy management system, the hardware composition should contain at least six main modules: data acquisition module, communication module, processor, data storage module, human-computer interaction module and power supply module. The concrete structure was shown in Fig. 2.

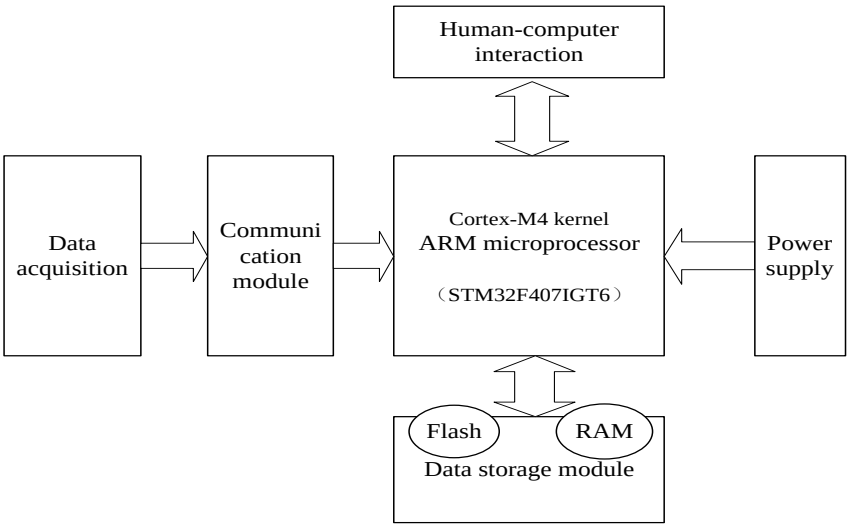

Fig. 2. System hardware architecture

The software structure of the whole system was shown in Fig.3, including the driving layer, OS layer, middleware, real-time database and application layer. The driver layer included a drive to display, serial port hardware parts and the board support package; OS layer used the embedded operating system μCOS-III to realize the multitask switch scheduling; The middleware was composed of a variety of API including the GUI interface; Real-time database was used to realize the fast data storage and query; The application layer was the basic task program for the realization of the function module.

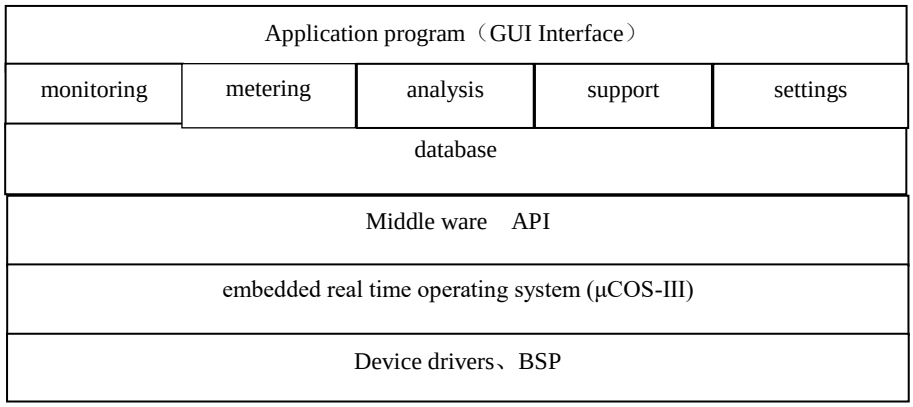

Fig. 3. System software architecture

3 Hardware design and composition

3.1 Hardware selection

In CPU selection, in order to meet the performance requirements of greenhouse energy management, considering the requirements of the use of embedded terminal platform and economic, the system took the STM32F407 micro-controller as the main processor. STM32F407 was based on advanced Cortex-M4 kernel and had enhanced DSP processing instructions and float point computing capability. It had up to

1M bytes of on-chip flash memory (flash) and 196K bytes of embedded SRAM, in addition, it had a flexible external memory interface which could extend NOR Flash, NAND Flash, SRAM and so on. The main frequency was 168MH, this running speed could achieve the processing ability of 210DMIPS which could meet the demand of the design.

At present, most of the other areas of energy management communication using RS485 interface and Modbus communication protocol, and most energy metering devices supported this communication, therefore, the design used this kind of quite mature communication mode. This protocol supports the traditional RS-232, RS-422, RS-485 and Ethernet equipment, in addition, it also had the advantages of standard, open, simple and compact frame format, etc.

Storage module was mainly used for temporary data storage and power down protection of data storage. RAM used to achieve the temporary data storage, was expected to need 1M storage space, and the model of this design used is IS61WV102416BLL-10TL, which could meet the requirements of the data storage module. Flash was used for the preservation of data after power outages and storage of data statistics, in the selection of Flash, NAND Flash had a large capacity, the memory could meet the code requirements, and it also had the advantages of fast rewriting speed, low cost and practicability, so, this design selects NAND Flash, the type was HY27UF081G2A.

In human-machine interaction selection, RA8875 was a powerful and low-cost color TFT controller, and it provided low-cost 8800/6800 parallel MCU serial port. In addition, RA8875 could combine text and 2D graphics applications, it also had built-in touch-screen function, and It could support the 800*480point resolution of small and medium size digital panel, therefore, the LCD screen used the RA8875 driver in this design and in order to better human-machine interaction experience, the design choosed 7 inches of liquid crystal display.

About power supply, the system directly used an external DC power supply adapter, the voltage range of 7-32V DC and the power was not less than 10W. It was also suitable for the STM32-V5 development board which we used to verify[12].

3.2 Data acquisition

In the acquisition module, compared with other areas of energy management system, the energy management system in greenhouse environment had a requirement of wide range and high accuracy for the data collection. In addition to using various sensors to get accurate detection of real-time temperature and humidity, light intensity, carbon dioxide concentration and other environmental variables, it needed to accurately collect water, electricity, heat, gas and other energy consumption data. In the acquisition of energy consumption data, the system not only needed to collect a variety of energy consumption, but also to the specific refinement of different equipment, different periods of energy consumption. At present, energy measurement mainly rely on flow meter, smart meters and other measuring devices to achieve data acquisition, the specific collection of the situation was as shown in Table1.

Table 1. Data acquisition in greenhouse

number	type	device	position	strategy	device model
1	water	flow meter	irrigation / wet curtain/ water supply pipeline /etc.	directly measure	GLP/TDS-100 series ultrasonic water meter
2	electricity	electric meter	fill-in light /fan/ shade curtain/	directly measure	SDT670 series electric meter

			water pump / state grid/etc.		
3	heat	meter/sensor	hot water pipeline/ hot air heater/ air conditioner/ etc.	Indirect calculation	GLP/TDS-100 series ultrasonic heat meter
4	gas	flow meter	gas pipeline	directly measure	GLP/TDS-100series
5	temperature and humidity	temperature and humidity sensor	Inside and outside the greenhouse	distributed acquisition	SHTxx series/ DHT11
6	CO2 concentration	CO2 concentration sensor	Inside and outside the greenhouse	distributed acquisition	T6713 series
7	light intensity	light intensity sensor	Inside and outside the greenhouse	distributed acquisition	ISL29013 series

4 Embedded software design

4.1 Transplantation of embedded system µCOS-III

µCOS-III source code can be divided into the documents related to computer hardware, the files related to the application and various service files related to system kernel. At the time of transplantation, it was necessary to modify the files related to computer hardware such as OS_CPU.H file, OS_CPU_A.ASM file and OS_CPU_C.C file. And the documents of system kernel such as OS_FLAG.C, OS_CORE.C, OS_MBOX.C, OS_MUTEX.C, etc. and the documents related to application as INCLUDES.H and OS_CFG.H did not need to modify.

In the OS_CPU.H file, we needed to set some parameters: 1) the use of the data type, that is, the unified expression of the length of the data unit; 2) stack parameter settings, such as the operating unit, the growth direction, etc.The stack growth mode supported by STM32F407 is from top to down, that is, the stack space is from high address to low address growth; 3) task interrupt macro definition options: Mask all interrupts, restore all interrupts used to ensure that the important function or task running data (code critical section) will not be changed and task switching is used for task switching in µCOS-III system.

In OS_CPU_C.C file we needed to use the C language to write the task stack initialization function: OSTaskStkInit (). This function was called by OSTaskCreate () or OSTaskCreateExt () when the task was created to initialize task stack, and it was closely related to the characteristics of the processor.

In the OS_CPU_A.ASM file, we needed to use assembly language to write the underlying function: OSStartHighRdy (), OSCtxSw () and OSIntCtxSw (). OSStartHighRdy () was called when the multi-task system starting the function OSStart (), it setted the run flag bit of system OSRunning = TRUE and loads the stack pointer of the highest priority task in the ready list into the SP and force it to return; OSCtxSw () was called in the task machine switching function, the task level switch is realized through the SWI or the TRAP man-made interrupt, ISR's vector address must point to OSCtxSw (); OSIntCtxSw () was called in the task exit interrupt service function OSIntExit (), to achieve the interrupt level task switch. Because it was in the interrupt call, the processor registers into the stack had been completed, we just needed to adjust the stack pointer.

4.2 Transplantation of SQLite database on μCOS-III

SQLite database operating system interface achieved an operating system abstraction layer, was designed to be compatible with a variety of operating system platforms, and it was composed of three subsystems: mutex, memory allocation and virtual files[13]. Memory allocation subsystem was used to realize the allocation and recovery of memory by SQLite; The mutual exclusion signal quantum system was used in the multi thread environment to use the linear SQLite resources, and the creation of the signal quantity needs to allocate the memory resources through the memory allocation subsystem; Virtual file subsystem was used in SQLite and the underlying operating system to provide a unified file operation interface, the creation of the file, delete and other operations also needed the support of memory allocation subsystem.

In multi thread environment, SQLite used the serial access to share resources in the system of mutual exclusion. Because μCOS-III supported multi task management, it was necessary to set the variable SQLITE_THREADSAFE=1 to enable SQLite support for multiple threads. Mutual exclusion was a kernel object that was defined as the OS_MUTEX data type, which was derived from the structure OS_MUTEX. μCOS-III allows users to call the mutual exclusion of the recursive signal to avoid the priority inversion problem, and thus the configuration could be set DSQLITE_HOMEGROWN_RECURSIVE_MUTEX=1 to enable the SQLite to support the recursive signal. In order to realize the mutual exclusion signal quantum system of the database, it was necessary to redefine the structure of sqlite3_mutex and used the signal operation function of μCOS-III to achieve the package of the signal operation function of SQLite database.

SQlite provided dynamic allocation and static allocation of mutex two mechanisms, and there are 8 types of mutex (2 kinds of dynamic, 6 kinds of static). To achieve dynamic allocation of mutex, we needed to use sqlite3MallocZero() function to create a pointer to sqlite3_mutex, and used the OSMutexCreate() function provided by μCOS-III to create the OS_MUTEX object. For statically allocated mutex, we could create a static sqlite3_mutex type array to store.

Because the μCOS-IIIwas a small and medium size embedded system, it did not provide a file system, however, it had a good extension performance, users could add the file system on their own. Embedded file system μC/FS was designed for resource limited embedded applications, it used the modular structure, providing a variety of modules for different hardware configuration [14]. And it provided an interface to support μCOS-III operating system by default, which could be transplanted to the system.

4.3 Transplantation of μC/GUI

μC/GUI is a graphics support system in embedded applications [15], it is designed to provide an efficient graphical user interface independent of the processor and LCD controller for any application that uses LCD graphics display. It applied to a single task or multi task system environment, and was applicable to any size of the true display or virtual display with any LCD controller and CPU [16]. As was shown in Fig.4, the software architecture of μC/GUI was modular, composed of different layers of different modules. The bottom layer directly points to the LCD hardware operation, and different systems need to do the corresponding transplant based on different LCD controller. The second layer was the most important level of the LCD driver, it used the hardware operating layer to achieve the most basic drawing function. The third layer was the package of the function library to achieve a complex graphics functions, providing API interface to the user layer to solve the problem of most of the problems occurred in the drawing. Users could add their own applications to achieve the graphical interface system by API interface.

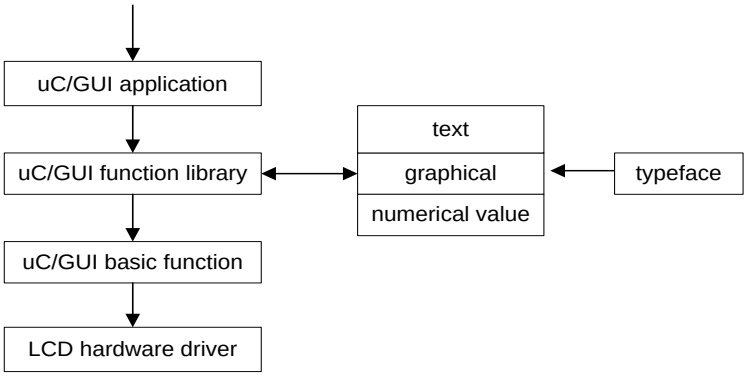

Fig.4. μC/GUI software architecture

The transplantation of uC/GUI mainly includes:

1) The definition of multi system support, window management and so on in guiconf.h file.

2) Hardware interface configuration and definition of the bus interface and register interface in lcdconf.h file.

3) The essence of LCD driver programming was corresponding to the point on the LCD screen programming and the bottom function calls for painting point function, it operates according to its own platform, bus interface and register interface or LCD controller register. First of all, we wrote the operation function of graphical display of the the underlying drivers such as the setpoint () and GetPoint (); Then, we realized the basic drawing functions such like picturesque circle, draw lines and frame by GUI_Line(), GUI_Rectangle() and so on; Finally, we setted the interface between the bottom driver and the uC/GUI function, because there was the basis of the above, we just need to fill in the interface function.

After the completion of the above work, we only needed to write the interface file between uC/GUI and uCOS- III, touch screen files, that is, to modify the GUI_X_Touch.c and GUI_X_uCOS.c files to successfully run uC/GUI on uCOS- III system.

4.4 Software process design

The entrance function after the system powered up was the main function, at the beginning, it needed to complete the initialization of the hardware, the operating system initialization and the creation of tasks and so on. We completed the system initialization directly using the system function OSInit (), and the hardware peripheral initialization mainly includes:

```
RCC_Configuration();   //System clock initialization and port peripheral clock enable
NVIC_Configuration();  //Interrupt source configuration
IO_Configuration();    //IO port initialization
TP_Configuration();    //Touch circuit initialization
FSMC_LCD_Init();       //TFT FSMC interface initialization
```

In the creation of the tasks, we directly used the OSTaskCreate () function provided by the system to create tasks and the main tasks mainly include: interface display task, touch operation task, data processing task, communication task and some other tasks like alarm. After the establishment of the above tasks, the system performs OSStart () function to start the multi task environment, so as to carry out the multi task management and scheduling. The operation of the system was as shown in Fig.5.

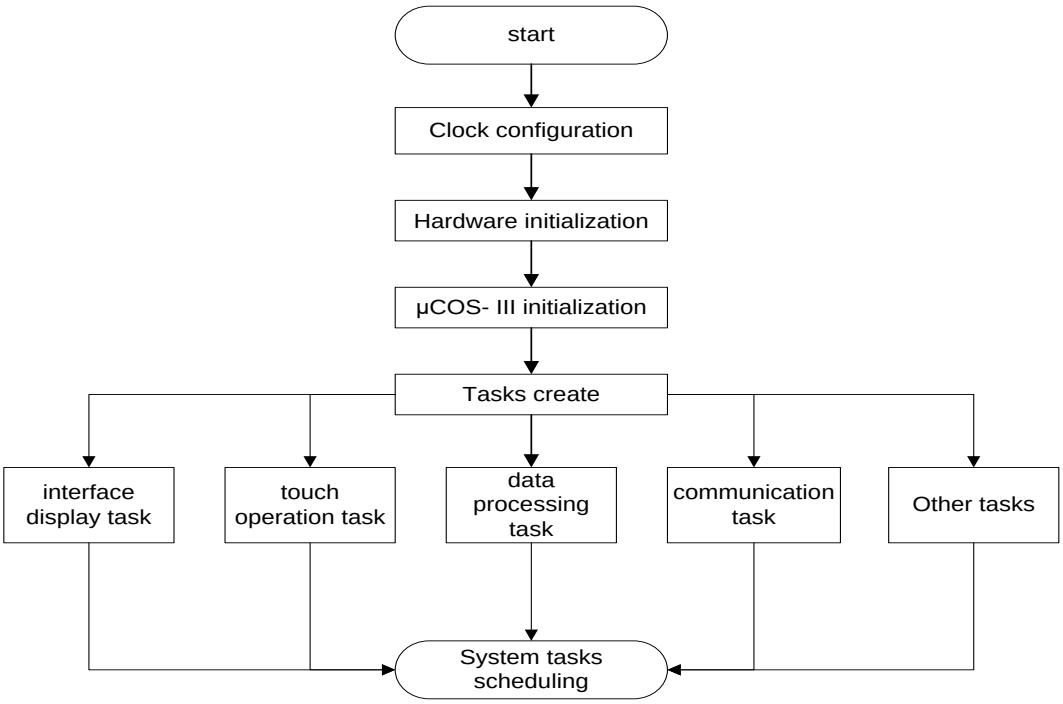

Fig.5. Main flow chart of system software

The system was based on priority to perform the task scheduling, which assigned different priorities according to different tasks. The whole process of the system was as follows: The platform first completes the communication with each intelligent instrument to receive the data, then the data was analyzed and the results are displayed in the interface so as to complete the human-machine interaction task. Therefore, the priority ranking should be communication task, data processing task, interface display / touch screen operation tasks (human-computer interaction task). Due to the other tasks such as alarm interrupt was a random event, the priority is relatively low.

5 System verification

The whole platform was verified in the STM32-V5 development board of Armfly company and the user login interface is as shown in Fig. 6.

Fig. 6. User login interface

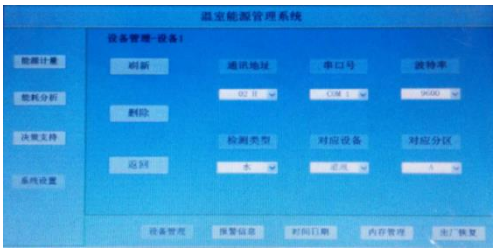

Fig. 7. Device configuration interface

When the platform was connected to a new metering device, the user needs to configure a series of parameters such as the serial number, address, baud rate, corresponding energy consuming equipment and energy sources, etc. The specific interface was as shown in Fig.7.

In energy metering, the user can choose measurement range by the buttons of lower side, and then set the metering time, energy and graphics types through the drop-down menu so as to inquires the energy consumption, the specific interface was

shown in Fig. 8.

Fig. 8. Energy measurement interface

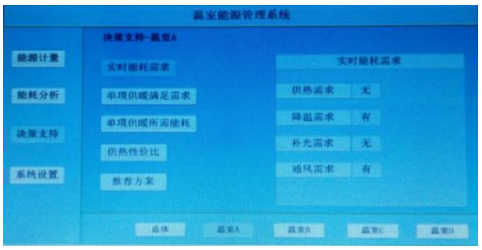

Fig 9. Decision support interface

In decision supports, the system could be extended to give different support decisions according to different conditions of greenhouse. As showed in Fig. 9, the heating / cooling demand of greenhouse, the price performance of greenhouse heating equipment and some other energy saving supports were given.

6 Conclusion

The design selected the suitable hardware to establish the embedded greenhouse energy management platform, in order to complete the basic functions of energy management. The real-time operating system μCOS-III was used to realize the switching between multiple tasks, the SQlite database was used to achieve the fast data processing, and the human-machine interaction interface was designed based on μC/GUI. Because the system was an embedded platform with the advantages of low cost, convenient operation and other advantages, it also had a strong expansibility. After verification, it is proved that the platform can effectively meet the basic needs of energy management in the greenhouse, and it has a certain feasibility in the greenhouse industry in China.

Acknowledgment

Research was supported by the National Key Technology R&D program "Research and application of intelligent management platform for greenhouse energy" (2014BAD08B0202), Beijing Academy of Agriculture and Forestry Sciences project(KJCX20170204) and the Construction project of Beijing Engineering Laboratory of Agricultural Internet of Things (KJCXPT2018-25).

References

1. You Zhangjin. Research on energy management system of manufacturing enterprise based on BACnet protoco. Beijing: Beijing Institute of machinery industry automation, (2010)
2. Wei Ayong, Ling zhihao, Pan Mengmeng. Development of building energy management system. Journal of Shanghai Dian Ji University, 2013,16 (3): 141-145
3. Kintner-Meyer M,Conant R.Opportunities of Wireless Sensors and Controls for Building Operation. Energy Engineering Journal, 2005,102(5):27-48
4. Zhang Yanyu, Zeng Peng, Zang Chuan-zhi. Summary of research on home energy management system in smart grid environment. Power system protection and control,2014,42 (18): 144-151
5. Jin Xiaogang, Ye Zhou, Zhang Shen-ming. Research on energy management of intelligent building. Application of energy technology, 2010 (10): 48-50
6. Menzel K, Pesch D, O，Flynn B, et al. To-wards a Wireless Sensor Platform forEnergy Efficient Building Operation. Tsinghua Science and Technology,2008, 13(S1): 381-386
7. FengYanping，Wu Yong，Liu Chang-bin．Energy—efficiency supervision systems for energy management in large public buildings：Necessary choice for China．Energy

Policy, 2009, 12(33):2060-2065

8. Zhang Quan. Application of building automation in green building. Electrical engineering, 2012 (7): 9-12
9. Cui Ran, Ma Xu-dong, Peng Chang-hai, Sun Yu. Design and implementation of building energy management system. Computer technology and development, 2010,20 (7): 184-187.
10. seafreak0123. Argus Provincial energy and PRIVA computer control technology.http://www.docin.com, 2012-02-28
11. Deng Lujuan. Research on model and control strategy of intelligent greenhouse. Shanghai: Shanghai University, 2004.
12. Armfly company. Armfly STM32-V5 development board user manual V2.0.
13. Hipp DR. Custom Builds Of SQLite or Porting SQLite To New Operating Systems. http://www.sqlite.org/custombuild.html.2013-06-26.
14. Li Jia-liang, Mu De-jun. Research on the transplantation of μC/FS file system based on SD card. microprocessor, 2010, (6): 79-81.
15. Wang Lanying. μC/GUI transplantation and realization of embedded system based on STM32. Journal of Sichuan University of Science and Engineering: Natural Science Edition,2012,25 (1): 56-58.
16. Feng Zhinian, Wu You-yu. Design of embedded GUI based on μC/GUI. Journal of Wuhan University of Technology,2006,28 (z1): 503-509