Implementation of Pan-Tilt System for Locating Based on ARM

Xu Shuping
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710032, China
e-mail: 563937848@qq.com

Zhang Zhiyong
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710032, China

Chen Yiwei
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710032, China

Li Hao
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710032, China

***Abstract*—The purpose of this paper was to implement a cheaper Pan-Tilt System than the traditional one which based on servo. The hardware platform consisted of the module of locating perception, the s3c2440 controller and the driver of stepper, and the Pan-Tilt System could stabilize at positions at arm/Linux circumstances controlled by software we programmed. The Pan-Tilt System had higher stability of dynamic under the condition of low frequency of pulse or high subdivision, and higher precise under high subdivision. This solution of Pan-Tilt System, suited for the place of low speed or loose real time, owns values of use and reference. The strategy was put forward for improving.**

Keywords-Stabilized Platform; PID Control Algorithm; ARM Controller

I. INTRODUCTION

Recently, it's common to use PTZ cameras in many fields. The application of the inspection robot has solved the problems caused by the lack of human or unattended substations in a certain extent[1-3]. The UAVsare applied to the transmission line inspection, which can greatly alleviate the increase of the total mileage of the transmission line, and the operation and maintenance difficulties caused by it can break the weak link of human inspection and save manpower [4]. Security monitoring provides an effective way for public security and judicial organs to prevent and crack down on illegal and criminal acts and maintain social stability. It is also important for promoting a harmonious society and safeguarding people's stability and life [5-7]. The key is that how to make the PTZ stabilize quickly.

In this passage, a two-degree-of-freedom Pan/Tilt System is designed and implemented, which adopts stepper motor. The stepper motor has lower cost, simpler circuitry and easier control than using a servo motor. By reading the position feedback of the attitude sensing module and directly controlling the rotation of the stepping motor, the pan/tilt is stabilized at the designated posture, and can meet the application requirements in low-speed or real-time applications. To further introduce the control process based on the stepper motor control curve to lay the foundation for the rapid and stable positioning of the PTZ[8-10].

II. SYSTEM ACTUALIZE PROJECT

A. Framework and Composition

As shown in Figure 1, the entire system consists of a controller, an execution module, and an attitude perception module. The solid line arrows represent the control output direction, which is the open-loop control mechanism of the stepper motor composed of the controller and the execution module. The dotted arrow represents the direction of the

DOI: 10.21307/ijanmc-2019-020

information flow and is a closed-loop attitude feedback mechanism composed of a controller and an attitude sensing module.

The pan-tilt system adopts a ring frame structure, and a stepping motor is arranged at both ends of the X and Y axes. The stepping motor is rigidly connected with the inner and outer frames through a coupling to ensure good transmission and real-time performance. The X-axis stepping motor is fixed to one end of the outer frame shaft, and the other end is fixed with a load of metal block to make it balanced. The Y-axis stepping motor is fixed on the base. The X-axis controls the tilt of the inner frame, and the Y-axis controls the outer frame axis to roll over.

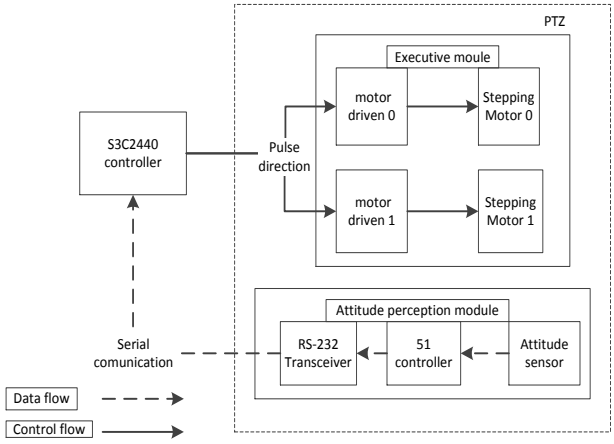

Figure 1. PTZ system frame diagram

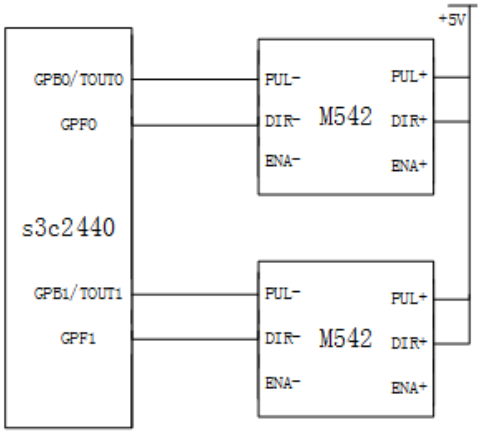

Figure 2. PTZ control drive wiring Diagram

In the system, 57HS09 stepper motor is used for X-axis and 57HS22 stepper motor is used for Y-axis. They are all mixed 2-phase and 8-wire. They are connected in series with the motor driver. They all use M542 stepper motor driver. The driver supports 15 The subdivision setting is dialed with a 4-digit dial switch, with a minimum of 400 and a maximum of 25600. A driver has two leads for receiving the pulse signal and the direction signal, respectively. However, it is required that the duration of the pulse high and low levels should be greater than 1.5us, that is, one cycle is greater than 3us and the maximum frequency is 330kHZ.

As shown in Figure 2, the controller uses a total of 4 pins connected to the stepper motor driver (M542). The GPB0 and GPB1 pins are used as Timers. The PWM function of Timer0 and Timer1 is used to form the pulse signal for stepping motor rotation. The GPF0 and GPF1 pins are set to output mode, and the output high or low controls the direction of rotation of the stepper motor. The P542+ and DIR+ of the M542 are connected to a +5V power supply. The enable signals ENA- and ENA+ are empty, and the stepper motor is always enabled.

The inner frame has a hollow cuboid shape (box body), the top is open and the inner and middle fixed attitude sensing module (XGZT-II) is composed of a 51 controller (c8051F350) and an attitude sensor as the core, and the controller outputs information. The sp3232 is converted into a level suitable for the rs232 interface as the output of the module. In response to the angle, the lower end of the box leads to two lines, one power supply and one serial port for outputting sampling data.

This article uses the FRIENDLY ARM mini2440 development board, which uses ARM920T-based s3c2440 controller, which is Samsung's product, is a widely used arm9 processor. The development board uses NAND Flash as a storage device.

B. Software Platform Composition

The Linux operating system has gradually grown from the 90s of the last century. It follows the design principle of "short and hard" and follows the GPL protocol, with its portability, cutability, security, stability, openness, open source, etc. It is widely used in embedded field.

Generally, a cross-compilation environment refers to compiling and installing a compiler that can be run directly on this platform to generate code for other platforms (such as ARM) on a PC-installed Linux platform. Here, arm-linux-(gcc, ld, objcopy, etc.) The tool chain is used to generate code that runs on the arm platform.Usually the embedded operating system needs to build bootloader, kernel, file system 3 parts.

The Linux kernel starts. In addition to the kernel image must be in the right place in the memory, the CPU must also have certain conditions [4], and these early preparations require the bootloader. This paper selects u-boot because of its status in the industry and its ability to support many hardware platforms and Linux operating systems.

The Linux kernel selected in this article is provided by the development board vendor (version 2.6.32.2), which provides good board-level support. The compiled kernel image can be used directly and is convenient. If you download from the official website, you must write the appropriate driver for the hardware platform and add it to the kernel source code.

The file system is a software organization responsible for managing and storing file information. In the embedded field, JFFS, YAFFS, UBIFS, etc. are more common. These are applicable to NAND Flash. This article uses the YAFFS file system.The file system is constructed by first creating a Linux tree directory under an empty directory (written as $path); building the busybox tool in a dynamically linked format and installing it in the $path directory; then through NFS. Debug, install the required software and libraries, write appropriate configuration files to achieve the required functions; Finally, use the mkyaffs2image tool to make the directory where $path resides as the YAFFS file system, which is recorded as rootfs.

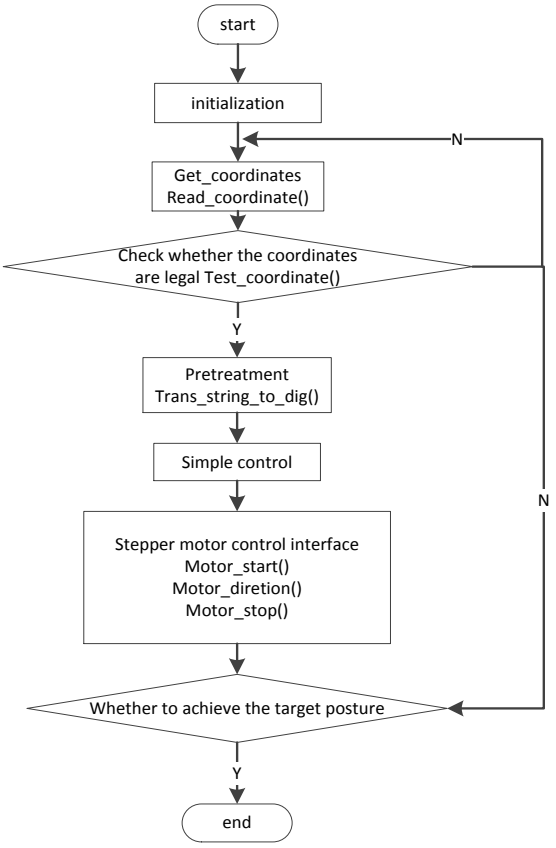

Figure 3. PTZ control software flowchart

III. STABLE PLATFORM SYSTEM CONTROL SOFTWARE

A. Data Acquisition

This module obtains angle information about X, Y coordinate strings from RS-232 interface (serial port). The attitude information data read out from the angle sensor module is a character string whose pattern is "X: - 00.000, Y: - 00.000", and the negative sign indicates the direction of rotation. The A function is used to read data from the serial port. Because the angle will change after the stepper motor rotates, the angle sensor data acquisition is set when the stepper motor rotates.

Since the collected angle information is in the form of a character string, each data packet is composed of a plurality of character strings, and it is necessary to check the integrity of the data packet to ensure the correctness of the collected data. If the collected data fails the test, then accept the data again.

Preprocessing module is used to convert accepted string data to attitude angle data. The trans_string_to_digit function is used to convert the pose data in a string to a float type value, which is returned in the form of an address. At this point, numerical data that can be used for attitude position feedback has been obtained.

B. Stepper motor drive module

As shown in Figure 3, the interface function is motor_*, and each function corresponds to a control command. There are 3 commands in total. Both have int fd and ioctl_data control parameters. Fd is the file descriptor, which refers to the device file; the ioctl_data type is defined as follows.

typedefstruct { intport; intfreq; int direction; } ioctl_data;

Port is used to specify the control port, T0 corresponds to the X-axis stepper motor, and T1 corresponds to the Y-axis stepper motor. Freq is used to set the transmit frequency of the PWM wave, if the frequency of the pulses received by the stepper motor driver. The direction is used to set the direction of rotation of the stepper motor.

Linux drivers can be used in two ways, compiled into the kernel and loaded modules. Compiling the kernel is a bit difficult and inconvenient, so the compilation into modules is more common.

A key data structure file_operations is defined in the kernel. As shown below, it defines the corresponding function pointer, which needs to be associated with the function that controls the hardware, to ensure that the hardware works properly when we call the open, close, and ioctl system calls.

structfile_operationspwm_fops=

{.open=pwm_open,.release=pwm_close, .unlocked_ioctl =pwm_ioctl };

For the stepper motor, the functions of starting, changing direction, and stopping need to be realized. In the pwm_ioctl function, the control logic of the stepping motor driving module program is implemented. The general structure is shown as follows.

```
staticintpwm_itcol(struct file *file, unsignedintcmd, unsigned long arg)
{
ioctl_data data;
  …
      //Check the validity of the parameters, if not legal return -EINVAL;
      //Copy user space data with arg address to kernel space with data address
switch(cmd){
case MOTOR_SPEED;            //Set frequency, start motor
  case MOTOR_DIRECTION;  //Change the direction of motor rotation
  case MOROR_STOP;         //Stop the motor
return 0;
      }
}
```

The entrance and exit of the kernel module. Themodule_init() and module_exit() macros are the entrances and exits of the kernel module. The parameters are function names, which are used to initialize the resource when the module is loaded (call dev_init) and release the resource when the module is unloaded (call dev_exit). Some of the codes and comments are shown in Figure 4.

```
Static int  __init dev_init(void)
{
     dev_init(...);
  //Initialize the device
     alloc_chrdev_region(…, “motor”);
  // dynamically allocate the device number, motor
  //is a string that identifies the device
  // and can be specified as another
     cdev_add(...);
  //Add a word device to the system
     return 0;
}
```

```
static void __exit dev_exit(void)
{
     cdev_del(...);
     //Remove device from system
     unregister_chrdev_region(...);
     //deregister the device number
}


module_init(dev_init);
module_exit(dev_exit);
```

Figure 4. DriverEntry

C. Hardware drivers

Call the stepper motor driver. Take motor_start as an example. It calls the ioctl system call interface, places the system call number in the system call with the R7 register, and then calls the swi (or svc) instruction to move from user mode to kernel mode.After the CPU responds to the interrupt, the PC jumps to the swi of the interrupt vector table and then jumps to the swi interrupt routine. Then save the site, take the system call number, take the system call table base address, jump to sys_ioctl. Sys_ioctl does not have a direct interface in the kernel. It is generated by SYSCALL_DEFINE3(ioctl, ...) when the kernel is preprocessed, followed by do_vfs_ioctl, vfs_ioctl, filp->f_op->unlocked_ioctl. Unlocked_ioctl is a function pointer to pwm_ioctl, which calls the code implemented in the driver.

D. System operation

Compile the code and generate two files: the application layer program and the stepper motor driver for the stable platform system control software, denoted as main and driver.ko. First copy the two files to the development board's file system/directory via NFS or scp commands. Then execute the following command in the development board Linux shell.

… .# cd /

… .#in smod driver. ko //Load the driver into the kernel

cat /proc/devices | grep motor // Look at the device number assigned to the driver, motor is the ID specified in the driver. Its output is:

253 motor //253 is the major device number assigned to the drive motor by the system and is used in the next command.

mknod /dev/motor0 c 253 0 //The device file motor0 is created. This is a Linux VFS application that is used to mask the underlying implementation and provide a uniform interface. 0 is the minor device number, and c is the character device.

./main //Running the program, stepper motor movement, the system can reach equilibri.

IV. RESULT AND ANALYSIS

Stepper motor drive fine fraction increase, it can obviously improve the stability of the system, but prolong the execution time, its pulse frequency is small, the output torque is big, it is advantageous to load ability, the stability degree is higher than the pulse frequency, however, too large and too little pulse frequency faces the problem of long execution time, and It is not reasonable to control the stepper motor directly with a given pulse, the characteristics of stepper motor can not be well played, resulting in the failure of stepping motor, overshoot, oscillation and so on. The

existence of these problems makes it difficult for the system to meet the demand of high real-time.

In order to reduce the execution time and increase the dynamic stability, it requires that the fine fraction of the drive is as small as possible when the precision is satisfied, and that the output torque of the stepper motor is large enough and the speed is fast enough, so it is necessary to study the step motor control process deeply. In the process of stepping motor acceleration, the torque is fully utilized to ensure the load and acceleration, so as to improve the running speed of stepper motor, reduce the execution time and improve the dynamic stability of the system.

TABLE I. IMPACT OF PAN-TILT BY SUBDIVISION

Motor drive fine fraction	Frame ring oscillation amplitude.	Find appearance time
2000	Little	Short
10000	Little	Middle
20000	Little, hard to see.	Long

TABLE II. IMPACT OF PAN-TILT BY FREQUENCY OF PULSE

Impulse frequency	Output torque	Carrying capacity	Frame ring oscillation amplitude.	Find appearance time
3125	Large	Powerful	Little	Long
31250	Middle	Middle	Middle	Middle
312500	Little	Powerless	Long	It's very long, and it swings back and forth across the end.

V. CONCLUSION

This passage designs and implements a 2-DOF PTZ attitude-finding system based on arm/Linux. The hardware is an open-loop actuator composed of an s3c2440 controller, a stepper motor driver, and a stepper motor, and a closed-loop attitude feedback mechanism formed by an s3c2440 controller and an attitude sensing module. The software is a PTZ system control program that includes data acquisition, algorithm control, and stepper motor drivers in a Linux-based cross-compilation environment.

In order to improve system performance, strategies can be taken: cropping the Linux kernel, removing unnecessary functions to improve the real-time performance of the system, studying the control curve of the stepper motor, and introducing corresponding control processes to improve the real-time performance of the system and the dynamic stability of the system; The existing PID controller is designed, improved and optimized for this system to improve the dynamic stability of the system.

ACKNOWLEDGMENT

The authors wish to thank the cooperators. This research is partially funded by the Project funds in shaanxi province department of education(17JK0381) and the Project funds in national university student innovation project (201710702006).

REFERENCES

[1] Melman S, Moses Y, Medioni G, et al. The multi-strand graph for a PTZ tracker[J]. Journal of Mathematical Imaging and Vision, 2017, 6(1):1-15.

[2] Liu N, Wu H, Lin L. Hierarchical ensemble of background models for PTZ-based video surveillance[J]. IEEE Transactions on Cybernetics, 2015, 45(1):89.

[3] Konda Kr, Conci N, De Natale F. Global Coverage Maximization in PTZ-Camera Networks Based on Visual Quality Assessment[J]. IEEE SENSORS JOURNAL,2016,16(16):6317-6332.

[4] Xu Y, Song D. Systems and algorithms for autonomous and scalable crowd surveillance using robotic PTZ cameras assisted by a wide-angle camera[J]. Autonomous Robots, 2010, 29(1):53-66.

[5] Yu JJ, Lu DF, Hao GB.Design and analysis of a compliant parallel pan-tilt platform[J]. MECCANICA,2016,51(7):1559-1570.

[6] Evren S, Yavuz F, Unel M. High Precision Stabilization of Pan-Tilt Systems Using Reliable Angular Acceleration Feedback from a Master-Slave Kalman Filter[J]. Journal of Intelligent & Robotic Systems, 2017(3):1-31.

[7] Mercader P, Åström K J, Baños A, et al. Robust PID Design Based on QFT and Convex-Concave Optimization[J]. IEEE Transactions on Control Systems Technology, 2017, PP(99):1-12.

[8] Li B. An optimal PID controller design for nonlinear constrained optimal control problems[J]. Discrete and Continuous Dynamical Systems - Series B (DCDS-B), 2017, 16(4):1101-1117.

[9] Kim J H, Hur S M, Oh Y. Performance analysis for bounded persistent disturbances in PD/PID-controlled robotic systems with its experimental demonstrations[J]. International Journal of Control, 2017:1-30.

[10] Saab S S. An optimal stochastic multivariable PID controller: a direct output tracking approach[J]. International Journal of Control, 2017:1-29.