*Advances in Intelligent Systems Research (AISR), volume 145*
2017 International Conference on Electronic Industry and Automation (EIA 2017)

# The Inversion of C/S Based on Protocol Firmata

Yongping HUANG*
Department of computer science and technology
Jilin University
Changchun, China
E-mail: hyp@jlu.edu.cn
+* Corresponding author

Fengyue YU
Department of computer science and technology
Jilin University
Changchun, China
E-mail: yufy15@mails.jlu.edu.cn

Danhui LI
Department of computer science and technology
Jilin University
Changchun, China
E-mail: 2501948885@qq.com

***Abstract*—In distributed control system applications, there are many different functions should be realized on upper computer and lower computer respectively. It increased the amount of software development, software upgrades and maintenance become difficult. By studying the firmata protocol, the inverted C/S request mode is implemented in this paper. The upper computer possessed the control logic and the lower computer realized the concrete behaviors, and the upper computer send request to the lower computer. With the inverted C/S request mode, the lower computers realized with the same program, and the different behaviors of lower computers are controlled by upper computer. Meanwhile, we program on upper computer in efficient Python, and it will reduce time for developing, simplify software deployment and improve abilities of upgrade and maintenance.**

***Keywords-python; protocol firmata; arduino; PID***

## I. INTRODUCTION

In most distributed control system applications, it is lower computer that builds a request while upper computer is just waiting in traditional client-server request mode [1] [2]. To get variety of functionality, it is necessary to program on each lower computer, and certainly, the specialized programming will increase software construction, also, requiring more and more time in upgrade and maintenance. Given this, this paper presents a study of firmata. We put control logic into upper computer, and let it make a request, then let lower computers accepting a request. That is, we will establish an inverted client-server request mode.

Firmata is a protocol used in the communication between host and embedded system, and it is well realized in the libraries of Arduino. Arduino is a famous and open source platform which is used in electronic development. Python is an interpreted and object-oriented language, having rich libraries [6].

With firmata protocol, all the controls and calculations are realized on upper computer, and there are only universal procedures on lower computers, no specific code. On host PC, we will write Python programs to control Arduino completely. It builds the bridge of Python programming with embedded intelligence developments. The applications realized in this paper are reading analog value and control servo [4] by PID [5]. Based on these, we can also extend multimachine controlling and multidevice controlling. That is very useful for the intelligent control and the embedded control.

The first part of this paper is the design scheme. The second part will give a detailed description of the concrete realization. The third part will make a test for the communication and show the effect. Finally, there is a conclusion.

## II. DESIGN SCHEME

This paper aims at completely controlling Arduino, with protocol firmata. We put some procedures into Arduino to make resolution protocol, and we write an applied program in Python on PC.

When we write Python programs on PC, there is no need for Arduino to have some extra operations, and we just need burn resolution protocol program which is named StandardFirmata. What the Python programs do firstly is to define firmata, secondly complete the application using the interface offering by Python-Firmata, and then we can send control commands to Arduino. This is the significance of firmata, separating the control logic and behavior. We don't need to consider configuration and principle of lower computers when making control on upper computer. The underlying code is generic, and the reason why lower computers can behave differently is that there are different procedures on upper computer. The flow chart of design scheme is shown in Fig. 1.

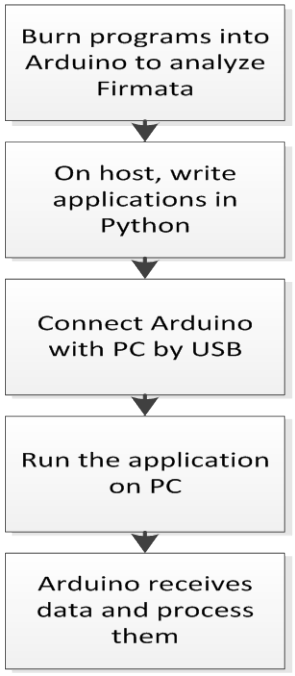


Figure 1. Flow chart of design scheme

It is just upper computer to decide what Arduino should do. We write application in Python, and implement protocol firmata also in Python, and then make communication with Arduino. I use the content published by tino in Github for a reference to implement firmata [7].

## III. Concrete Realization

In the previous section, we have given an outline of design scheme. There are two parts we should do. First, we may program to realize applications in Python. Second, we should burn StandardFirmata into Arduino, and then it can analyze instructions. The detailed operations are that Arduino receives analog signal, and let servo rotating through PID based on the analog value. The schematic diagram for connection is shown in Fig. 2.

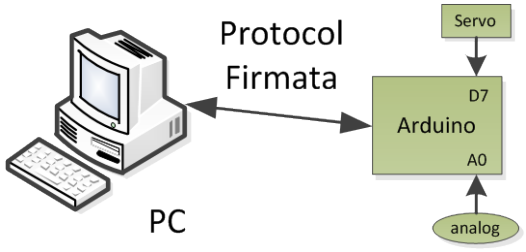


Figure 2. Connective schematic

### A. *Implementing Application on PC*

The main file is called pyFirmata.py. There are two main classes in it, class Board and class Pin. Board is used to create a Board instance. It has a function called get_pin used to define pins, while there is no need to call functions as digital or analog every time. There is a pin mode SERVO, it is used to control servo specially [9]. Pin is used to define pins. Actually, it is module serial to be used for communications. There are two functions in serial, function write and read. What serial does is shown in Fig. 3.

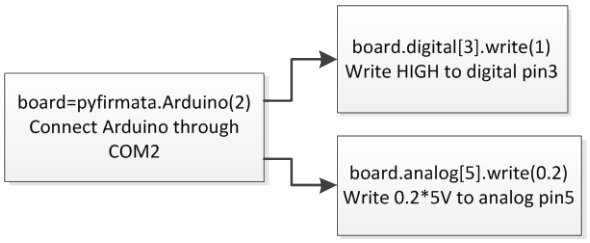


Figure 3. Things serial does

We should use Python to do like this, getting analog value from the analog pin, and convert the value into an angle which servo will rotate to [3]. There is a mode defined in firmata, SERVO mode [5], and it can be active on Pin 9 or Pin 10 to make servo rotating from degree 0 to degree 180. The treating processes are shown in Fig. 4.

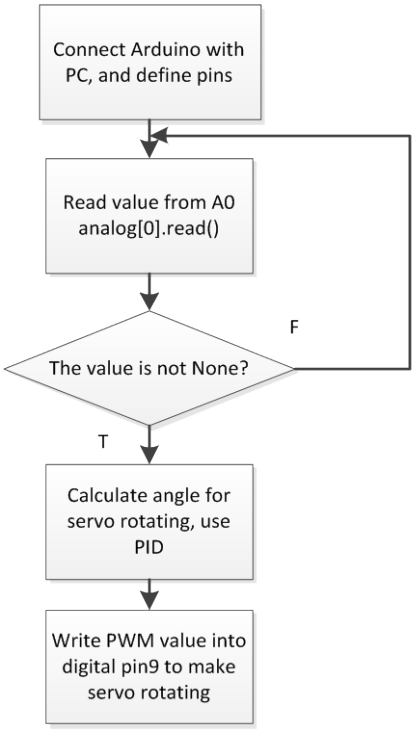


Figure 4. Treating processes

With firmata, we can write PID algorithm in Python and it just need to transfer data between Arduino and host computer. Power of PC is strong, so the algorithm will be implemented more quickly and easily on the lower computers. The incremental PID algorithm is shown in Fig. 5.

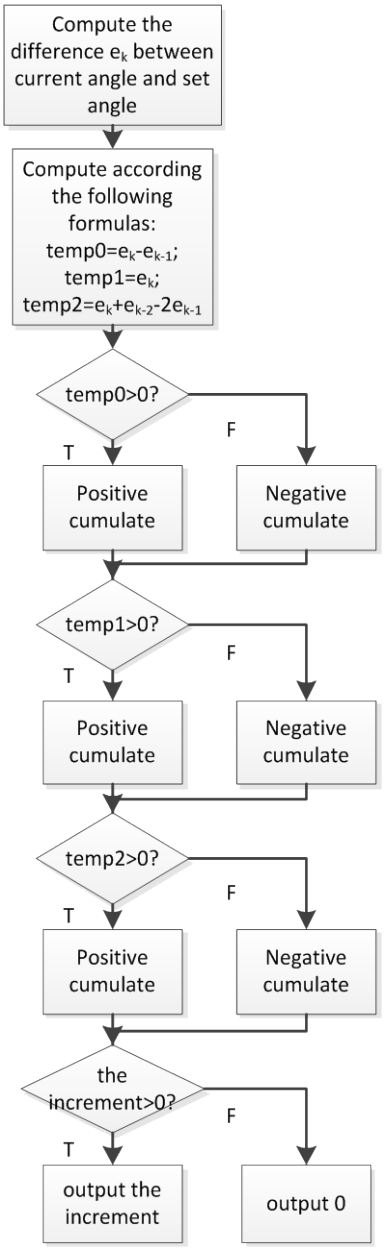


Figure 5. The incremental PID algorithm

The formula is $\Delta u_k = k_p * [e_k - e_{k-1}] + k_i * e_k + k_d * [e_k - 2 * e_{k-1} + e_{k-2}]$.

### B. *Processing Data on Arduino*

Using protocol firmata, there are some communication commands, such as DIGITAL_MESSAGE 0X90, ANALOG_MESSAGE 0XE0, SET_PIN_MODE 0XF4, START_SYSEX 0XF0, END_SYSEX 0XF7 [8], and there are also some callback functions attached to these command. According to these connections, Arduino can get data from PC. The connections are shown in Fig. 6.

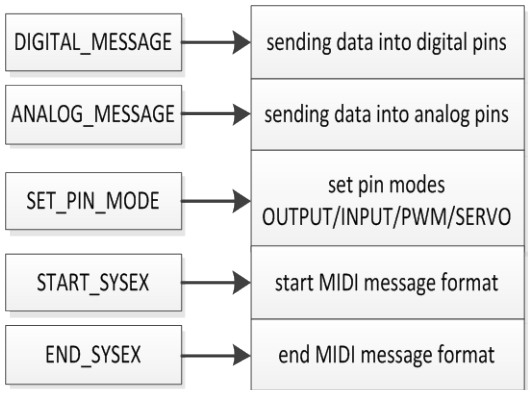


Figure 6. Commands and callbacks

There are two parts of communications, Receive and Send. Two main functions can be obtained in receive, available() and processInput(). Available is used to confirm usability of serial. ProcessInput is used to accept commands and process them. Send operation is used to transmit digital data or analog data.

All above is definitions written in Firmata.cpp, and then we should burn StandardFirmata into Arduino to analyze the commands sent from PC. AnalogWriteCallback and digitalWriteCallback are both used to write value onto pins. SetPinModeCallback is used to set modes of pins. We use function setup to initialize Arduino.

## IV. Test Procedure

Getting started, firstly connect Arduino to PC, and then run the Python program for testing.

The first item is on-off control. We can define pins on PC, output HIGH or LOW and light on or off LEDs connected on Arduino.

The second item is getting analog data from analog pins. Also, we define analog pins on PC, and then have a data collection. We use variable resistance to generate analog values [10]. It is just like Fig. 7 showing.


Figure 7. Analog value collections

And then control servo through digital pin9. We output controlled quantity and make servo rotating. It has been shown in Fig. 8.

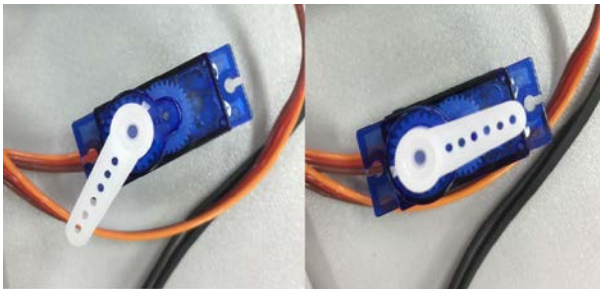

Figure 8. Servo rotating

We use PID control algorithm to output the controlled quantity, and control servo rotating through digital pin9. After the above testing, we have realized a wholly control on Arduino. There is only resolution protocol for firmata in Arduino, while all the control logic programs are written in Python.

## V. SUMMARY

Based on firmata, with the same program in lower computers, the handle procedures are different by upper computer control logic. This is the separation of control logic and concreted implementation. In this paper, we have realized on-off controls, PID control, steering gear control and data collection on slaves. It can increase efficiency for software development, simplify software deployment and be convenient for software's upgrade and maintenance.

What this paper has implemented is an inverted C/S request mode, it is a centralized control. Specific application depends on the logic control on upper software. However, because of the separation between logic and implementation, the instantaneity of the system relies on the networks. It is not suitable for the situations required highly real-time like industrial control. It is suited to be applied to internet of things and smart home, etc.

## ACKNOWLEDGEMENT

I really appreciate my tutor Huang Yongping who is a professor at Jilin University. He is an expert at embedded system. And then I would like to thank my parents for their support. Finally, I really appreciate tino in github, and I will never forget the contributions of many other people in Firmata.

## REFERENCES

[1] Daniel de Santos, Víctor Lorente, Félix de la Paz, José Manuel Cuadra, José R. Álvarez-Sánchez, Eduardo Fernández, José M. Ferrández. A client–server architecture for remotely controlling a robot using a closed-loop system with a biological neuroprocessor. Robotics and Autonomous Systems, Volume 58, Issue 12, 31 December 2010, Pages 1223-1230.

[2] P. Vařeková, I. Vařeková, I. Černá. Automated Computing of the Maximal Number of Handled Clients for Client-Server Systems. Electronic Notes in Theoretical Computer Science, Volume 260, 1 January 2010, Pages 243-259.

[3] Cai Ruiyan, Design of Servo Control System Based on Arduino, Computer Knowledge and Technology, 2012, 8(15) 3719-3721.

[4] Günter Spanner, Arduino: Circuits & Projects Guide, Elektor International Media BV, 2013, pp.221-225.

[5] Clemens Valens, Mastering Microcontrollers: Helped By Arduino, second ed., Elektor International Media BV, 2015, pp.134-137.

[6] Mark Lutz, Programming Python, fourth ed., O'Reilly Media, Inc., 2011.

[7] Information on https://github.com/tino/pyFirmata

[8] Information on http://firmata.org/wiki/Main_Page

[9] Information on http://lagunak.gisa-elkartea.org/projects/krnl/wiki/PyFirmata_servo

[10] Andreas Attenberger, Klaus Buchenrieder. An Arduino-Simulink-Control System for Modern Hand Protheses. Artificial Intelligence and Soft Computing, vol. 8468, 2014, pp.433-444, doi: 10.1007/978-3-319-07176-3_38.