ATLANTIS PRESS

Advances in Computer Science Research, volume 70

2nd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2017)

Design of control platform for Ballbot

Lei Liu [a], Rui Cao [b], Ming Liu [c], Chun Dong [d]

School of Electrical Engineering, Beijing Jiaotong University, Beijing 100044, China.

[a] liulei19920612@163.com, [b]caorui@bjtu.edu.cn, [c]15121438@bjtu.edu.cn, [d]chdong@bjtu.edu.cn

Keywords: Control platform, Ballbot, Physical modeling, Rapid control prototyping, Code generation.

Abstract. In this study, an experimental platform for observing the real-time performance of control algorithms is designed using rapid control prototyping. Details of the use of the platform, as well as the design and debugging process for the Ballbot controller are presented. The platform achieves seamless connection between the simulation and the actual system. This means that engineers can quickly and easily observe the true performance of their algorithm, which ensures an ideal research platform for control theory application and optimization.

1. Introduction

The Ballbot self-balanced control platform is a dynamic experimental platform based on a ball-type self-balanced robot designed using rapid control prototyping. It facilitates the development of robots and accelerates the process of debugging.

A Ballbot is a dynamically stable mobile robot designed to balance on a ball. Owing to its special power mechanism, a Ballbot is an omnidirectional mobile robot. It works on the same principle as an inverted pendulum. To guarantee its agility and maneuverability, the stability of the whole system is crucial.

The robot control platform is proposed to promote the efficiency of modelling and design of actual systems. Such a platform could seamlessly connect the simulation model with the actual system, so that engineers can easily design algorithms for simulation models and connect them to the actual system.

To implement the robot control platform and validate its functionality, the following work has been done in this study:

(1) The design of the hardware platform: we determine the layout for the electrical and mechanical subsystems necessary to manufacture the functional robot. The configuration of an electrical subsystem associated with the mechanical subsystem is introduced to establish a solid foundation of total system.

(2) The code generation framework: the driver code is wrapped by an S-function to make it available to code generation modules, thus forming a bridge between the MATLAB environment and the robot hardware [1, 2].

(3) The creation of the simulation model: we model the entire robot system through physical modeling. The physical model will contain the complete mechanical, electrical, and control parts as well as the actual system.

(4) Experiment: the real-time interaction between the control platform and the actual system is described in this section. In addition, comparison between simulation results and actual system behavior is made in order to verify the feasibility of our platform.

2. Electromechanical system configuration

A good electromechanical design is the basis for the proper functioning of an automatic control system. It involves a mechanical, electrical, and control component, with each part coupled to another, but all working relatively independently.

The overall configuration of the main body of the Ballbot is as shown in Fig. 1. It consists of a laser radar on the top that can explore the surrounding environment and build a map for navigation and planning purposes. The second layer is the main control subsystem, composed of a single broad computer, a wireless router and an inertial measurement unit. The third layer consists of two power lithium batteries. On the bottom of the main body, three drivers are assembled.

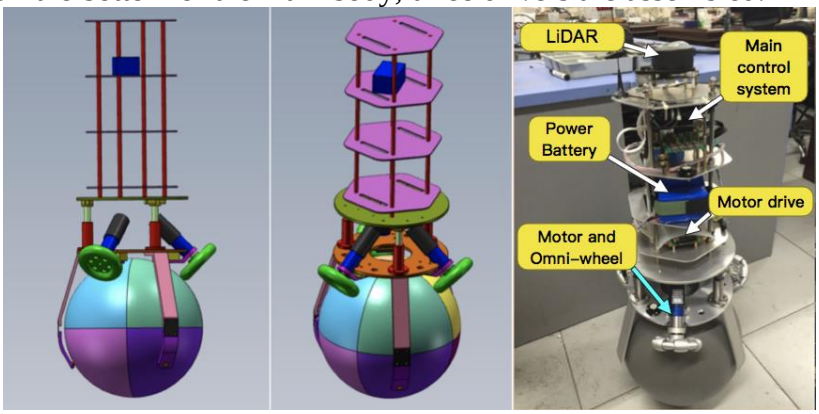


Fig. 1 Overall configuration of Ballbot

The electronic control system is mainly composed of sensors, actuators, and the controller (Fig. 2). The system communicates with the software platform on the PC through Wi-Fi to enable the wireless control of the robot. To obtain the current attitude of Ballbot, the 9-DOF sensor module is used. It provides both fused sensor data and raw data that can be handled by our own driver code. The electric drive subsystem uses the brushless DC motor 3564K024B and the matching motion controller MCBL3006S, which communicates with the CPU via CANopen. The single board computer is a Beaglebone-black open source platform (Debian 7.2) [3].

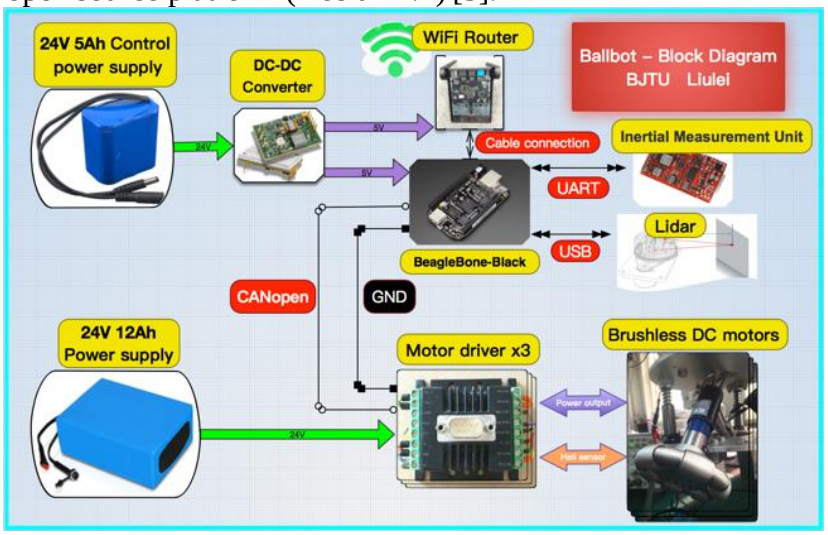


Fig. 2 Configuration of the electrical control system

3. Code generation module

Automatic code generation is the key to connecting simulation models to an actual system. MATLAB built-in blocks enable users to build their own algorithm. For the specific hardware driver, customized Simulink blocks are necessary to enable code generation [4]. These hardware driver blocks are packaged in the S-function [5]. This section is mainly concerned with the functioning of the sensor and motor block.

Fig. 3 shows the sensor code generation block without a mask. The output of the block is two buses: the calculated attitude angle and the original sensor data [6].

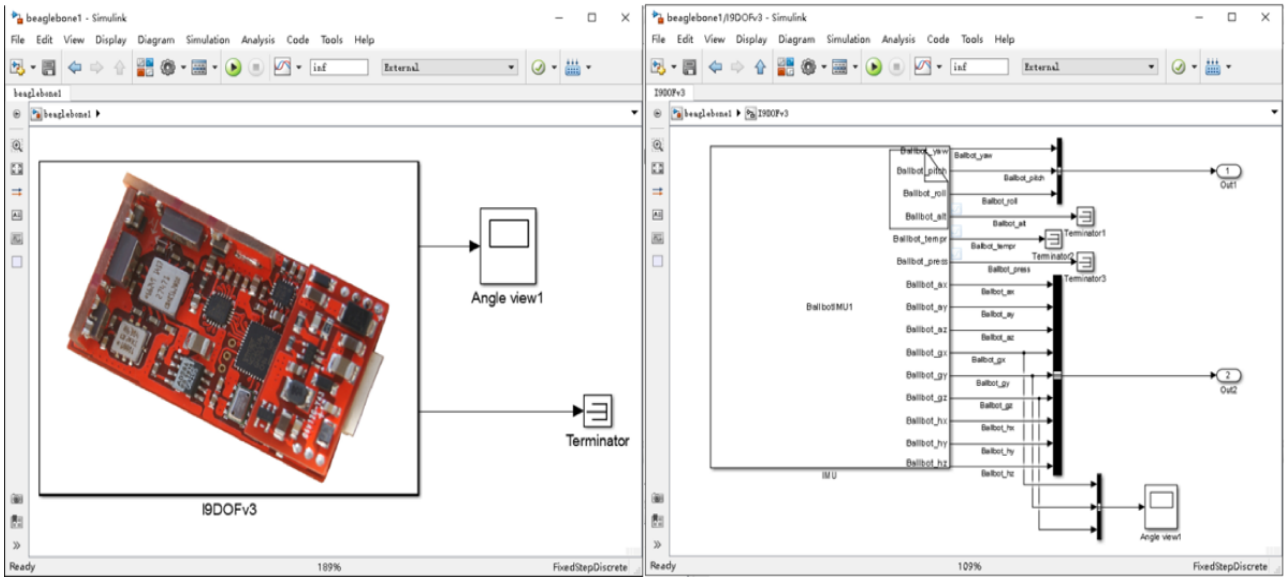


Fig. 3 Code generation module of the sensor

The complete motor subsystem model is shown in Fig. 4. The motor input is the target speed and the output is the current speed of the motor and the current signal. To facilitate system debugging, we added the GPIO driver. Through the key, we can initialize, enable, and disable the driver system. Additionally, the subsystem includes an LED to indicate its working status. Users can dynamically adjust every motors' parameters of the speed and position closed loop controllers.

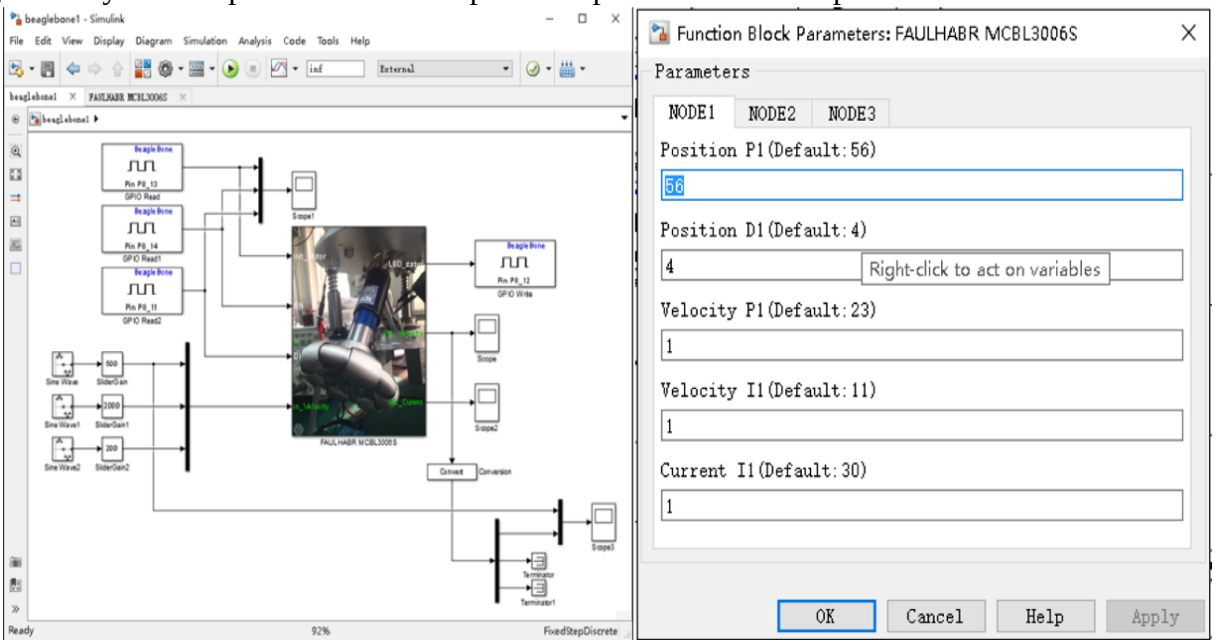


Fig. 4 General mask of the motor system

4. Physical modelling

Unlike obscure mathematical equations, physical modeling provides a natural and efficient method to construct a system model. Physical models reveal the composition of the system structure in a way that can be intuitively understood. Each module corresponds to an actual physical device, such as an oil pump, a motor, or an operational amplifier, so the controlled object is no longer just a transfer function. The physical simulation model of Ballbot is shown in Fig. 5.

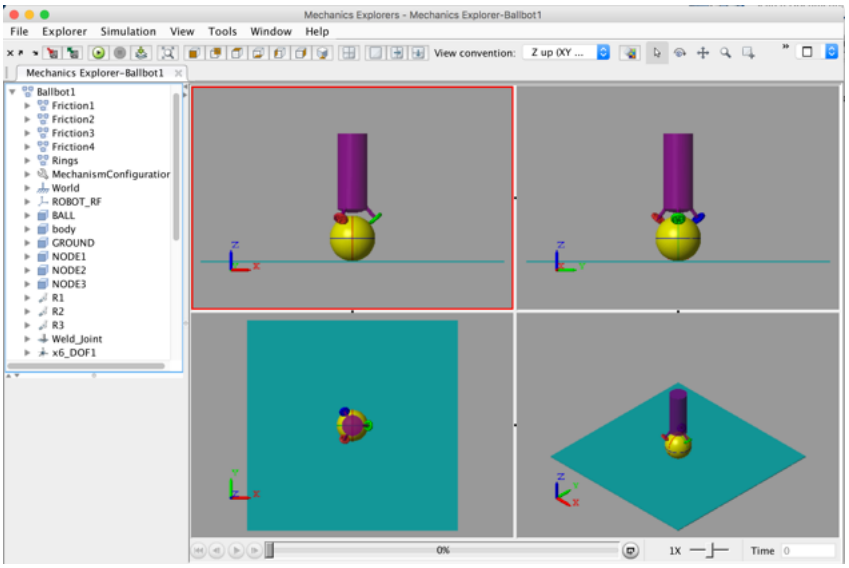


Fig. 5 Operation of robot system

The weight of each part of the physical model should be consistent to ensure that the inertia parameters of the model are correctly calculated. As shown in Figure 6, the total weight of the body is 8.565 kg, of which the omni-directional wheel weighs 0.275 kg and the ball weighs 2 kg. To design a model more similar to the real system, a driver must be incorporated.

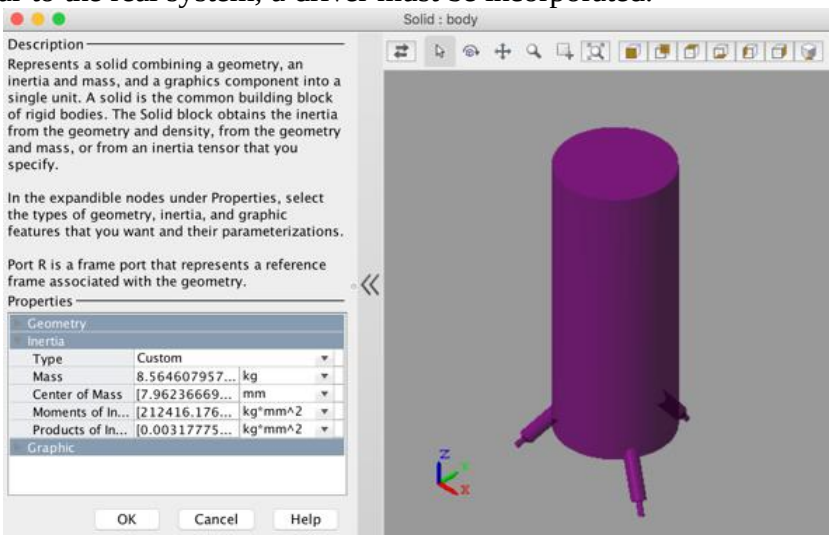


Fig. 6 Mass properties of Ballbot system

In the actual system, the motor and the driver form a subsystem. The motor model includes a closed-loop speed controller. The actual parameters of the motor selected in this paper are shown in Table 1. A complete electrical drive system simulation model (Fig. 7) includes motors, a closed-loop controller, a gear box and omnidirectional wheels. It converts electrical energy into mechanical energy, laying a solid foundation for the success of the entire simulation.

Table 1 Motor parameters

Rated voltage:24 V	Stall torque:371 mNm
No-load current:0.189 A	Rotor inertia:$34\ g \cdot cm^2$
Mechanical time constant:11 ms	No-load speed:11300 rpm

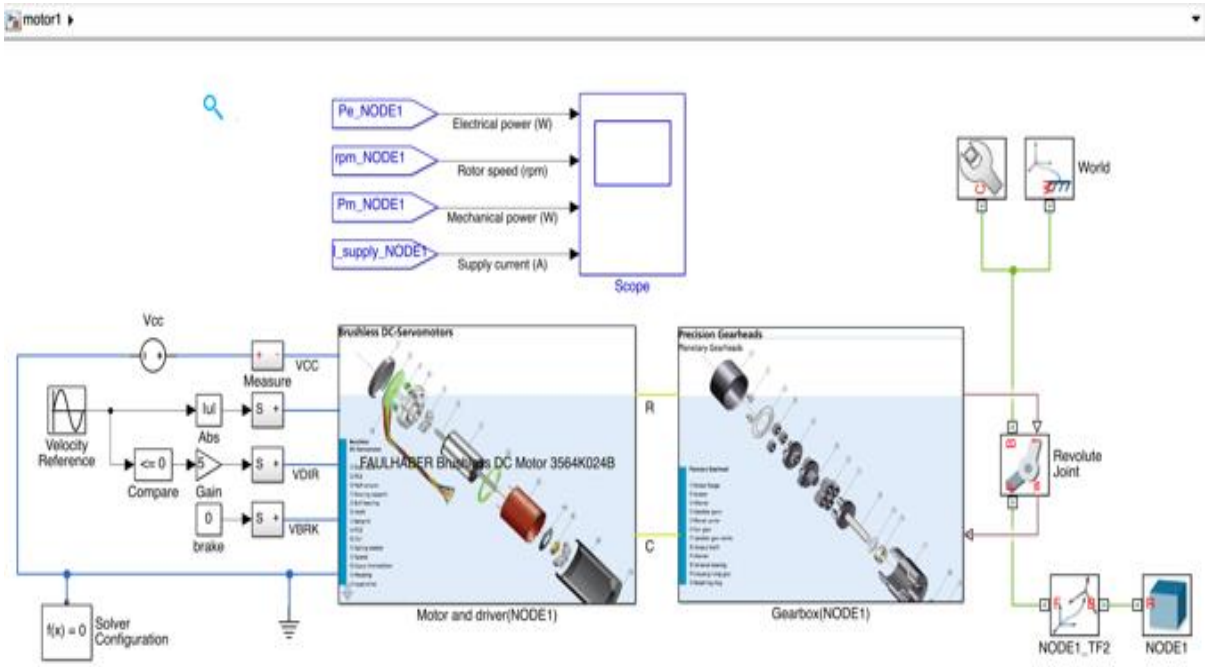


Fig. 7 Complete power drive system simulation model

5. Control platform

During system design, engineers should not only focus on control, but must maintain sight of the whole system [7].

After the algorithm is designed on the PC, it is run directly on the real-time target machine by automatic code generation. As a result of the seamless connection facilitated by the control system design software MATLAB, engineers can observe the performance of the control algorithm in real time and tweak the controller parameters. Further, they can compare simulation results and the actual response to accelerate the development process [8].

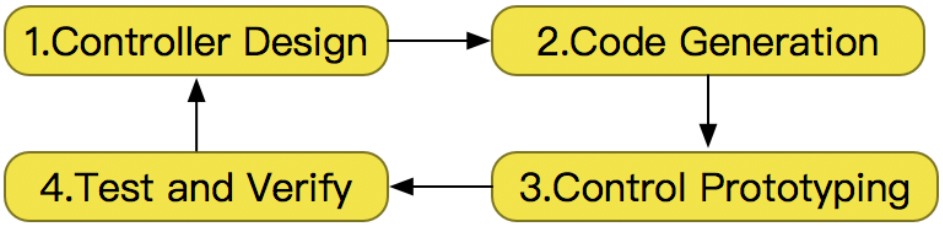


Fig. 8 Process for use of control platform

Once software and hardware design, and the simulation part of the control platform are complete, a practical workflow that can effectively combine theoretical simulation with the actual system must be determined [9]. As shown in Fig. 8, the development process is an iterative design of the loop, which can make the algorithm better by continuous iteration optimization [10].

Step 1: Design control algorithm for the robot simulation model in the Simulink environment.

Step 2: Replace the simulation model with the sensor and driver modules from the code generation library. Then, generate the code and download it to the real-time target in the Simulink environment.

Step 3: When the program starts running in the target machine, it sends real-time running data to the host computer through the wireless network. Engineers can observe robot status and adjust the controller parameters in real time through Simulink external mode.

Step 4: Using the data collected by MATLAB, we can compare the simulation results of the robot and the actual response. If control targets do not meet the requirements, then we return to the first step to optimize the algorithm.

6. Controller based on physical model

After the simulation platform is built, it is necessary to design a closed-loop controller to guarantee equilibrium and maneuverability. The complete robot simulation model is shown in Fig. 9. It is mainly composed of the body of the robot, the Euler angle module, the system input, and the controller.

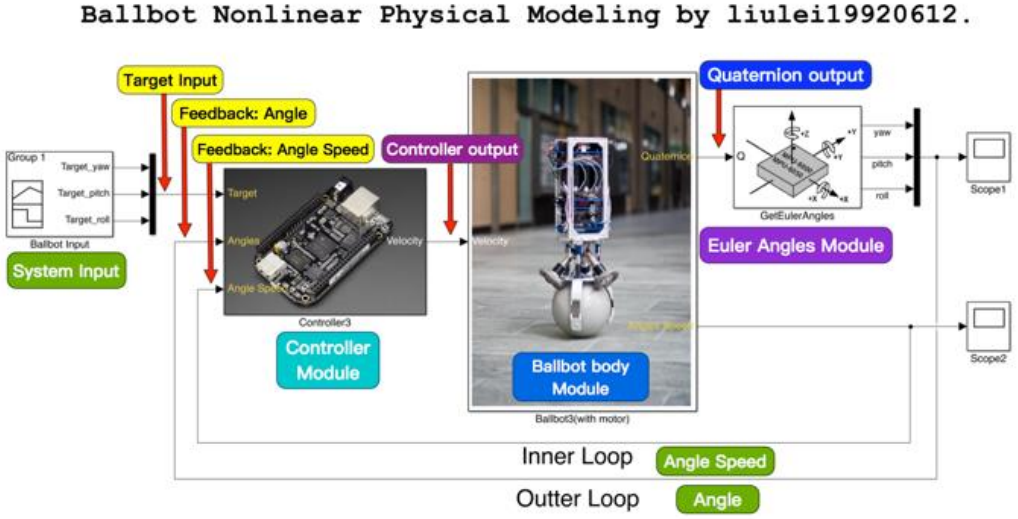


Fig. 9 Complete simulation of Ballbot system

The inputs to the controller module are the position reference signal, the angle feedback, and the angular velocity feedback, and the output is the motor speed signal. As shown in Fig. 10, the green box is the outer ring controller and the blue box is the inner ring controller, users only need to change their control algorithm within the module. The kinematic solution matrix can be used to alter the speed of each omnidirectional wheel by changing in the velocity in the reference coordinate system.

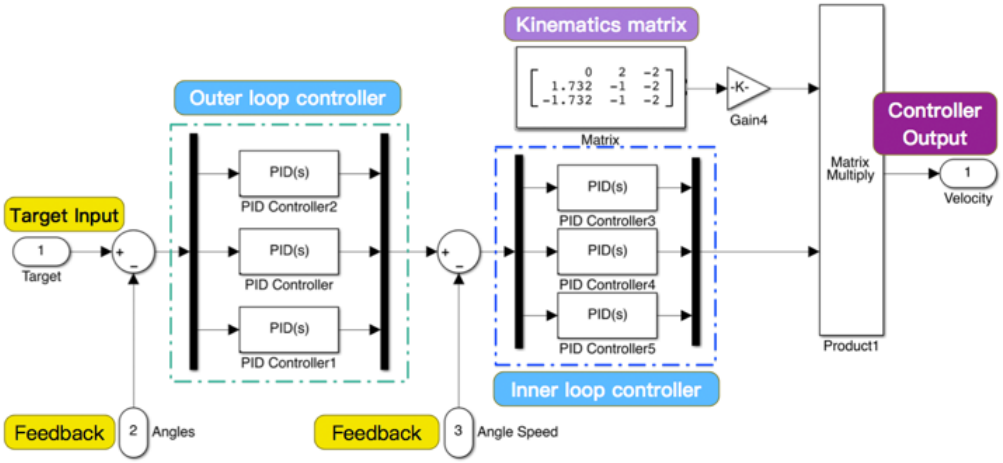
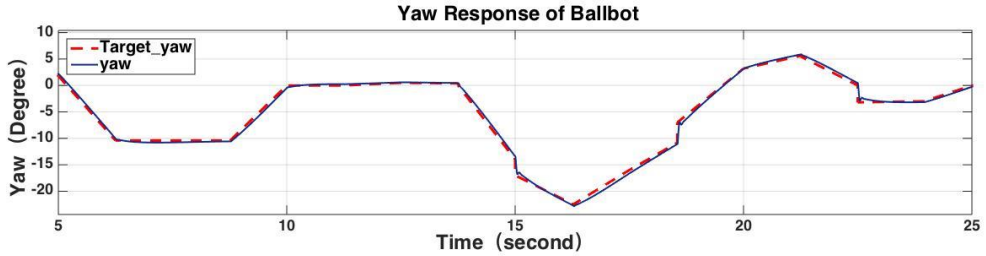


Fig. 10 Controller of Ballbot

To verify the performance of the controller, a series of signals is added to the input signals, in order to compare the input signal and the actual response. As shown in Fig. 11, the controller is still able to maintain the balance of the robot in complex situations. In general, the system response time is normal.



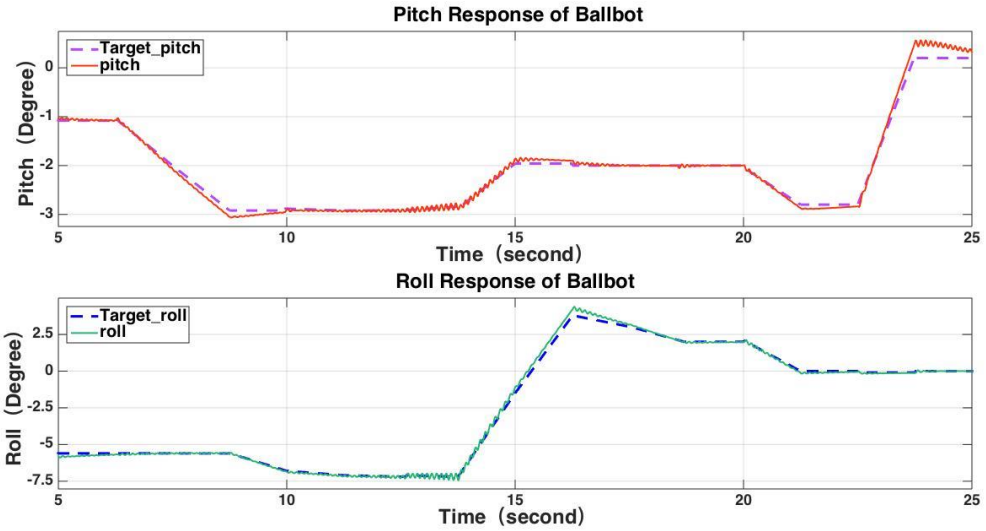


Fig. 11 Euler angles response of Ballbot

7. Experiments

The design process of the controller is based on the flow diagram shown in Fig. 8, and the first step of this process is done in Chapter 6. The second step, as shown in Fig. 12, involves transplanting the control algorithm.

Figure 6-1 shows that the structure of the physical controller, which uses a double closed-loop PID controller, which is consistent with what is used in the simulation environment. In Fig. 12, the inner ring controller whose control period is 5 ms is depicted using a green font and the outer ring controller whose control period is 20 ms is depicted using a blue font. The difference in control period has been optimized to match the actual controller.

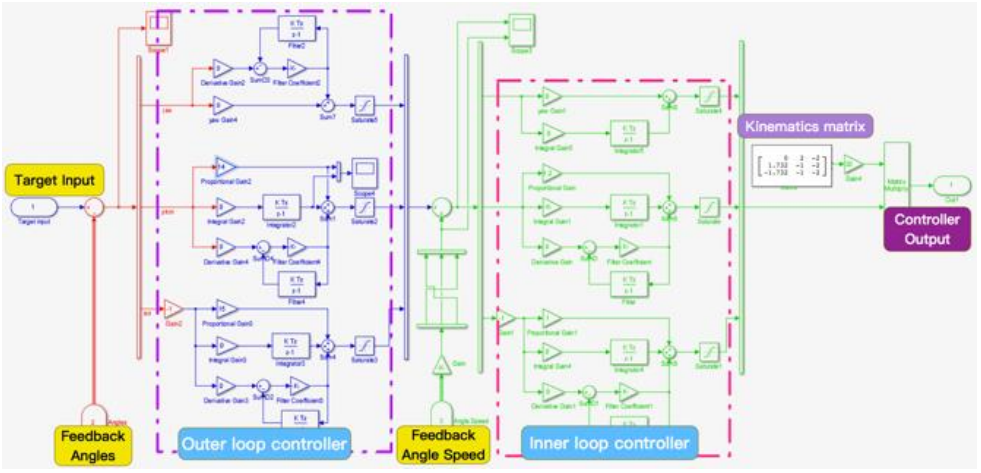


Fig. 12 Controller for Ballbot balance

The third step involves downloading the algorithm onto the target hardware during real-time operation. The actual control system of Ballbot is shown in Fig. 13. Engineers can transplant the algorithm without any code to be added to the model for the real system.

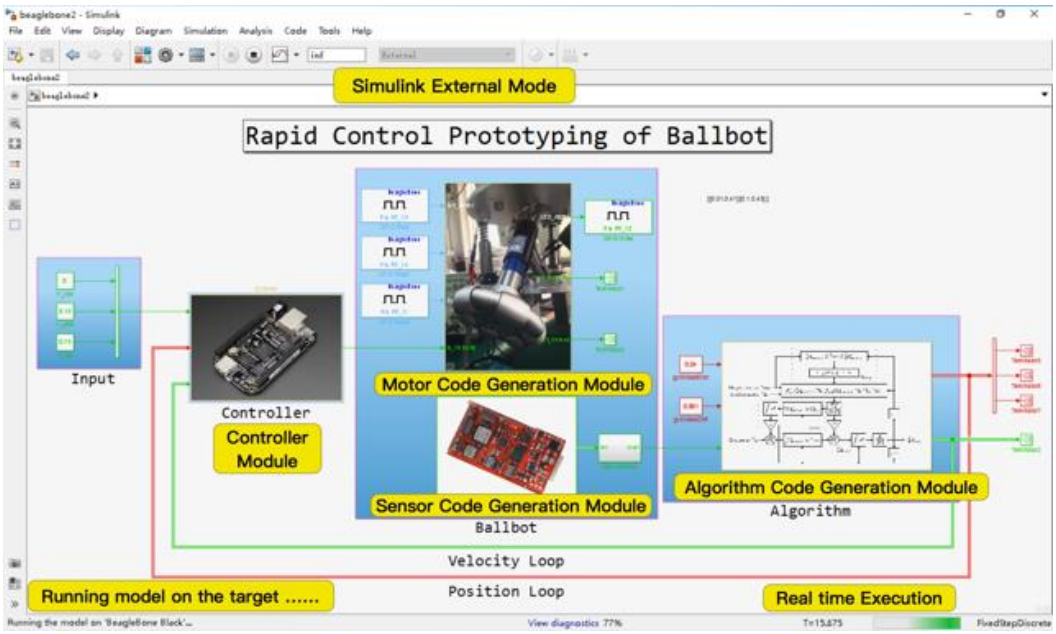


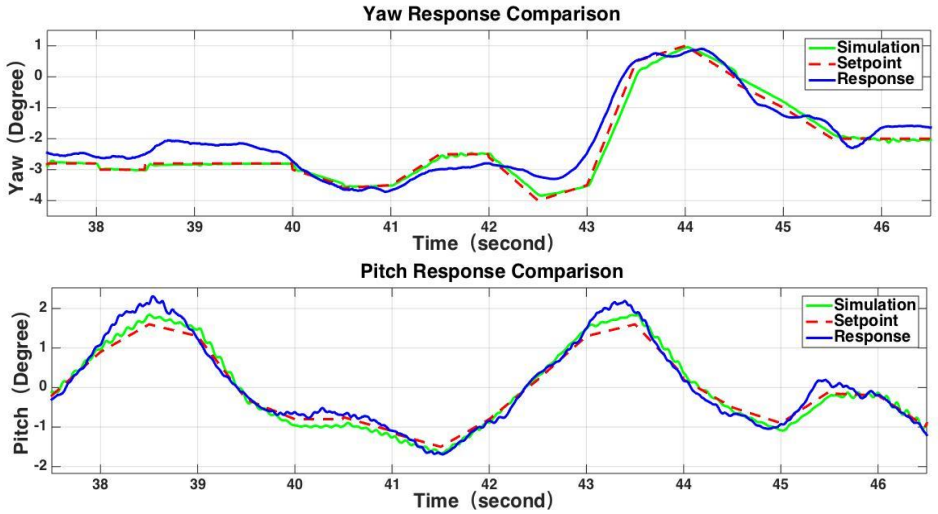
Fig. 13 Rapid control prototyping of Ballbot

The fourth step involves collecting the data and analyzing the differences between the simulation model and the actual system.

Series of signals are applied separately to the simulation model and the actual system. Fig. 14 shows the robot's attitude angle response, where the red dotted line represents the given signal, the green solid line represents the simulation results, and the blue solid line represents the actual response. The simulation results are very close to the given signal. The actual response cannot be followed by a given signal, but the overall trend is consistent with the simulation results, which is acceptable, considering the non-linear factors present in the actual system [11].

Throughout the entire operating curve of the system, the simulation results are in good agreement with the actual response. To improve the performance of the actual system, further optimization can be accomplished by means of rapid control prototyping.

The simulation results provide an intuition regarding the design aspect of the control system, and the experimental results help to determine the specific details. Through the analysis of the experimental results, it is found that the simulation model and the actual system are very different. The model cannot be completely accurate. To address this discrepancy, the simulation model may be built to understand the system characteristics. Subsequently, the specific parameters may be determined through the debugging process.



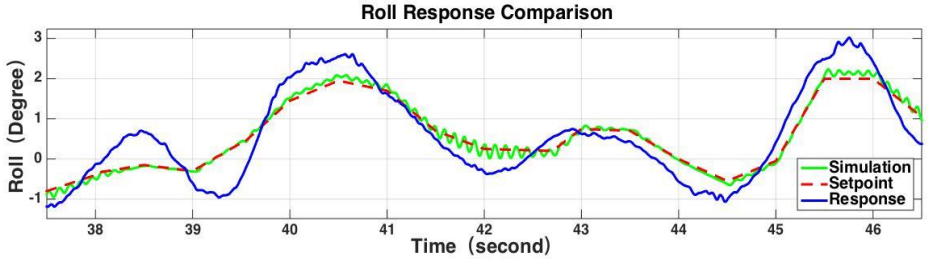


Fig. 14 Response of Ballbot attitude angle

The simulation results of the motor speed are shown in Fig. 15. Their amplitude and overall trend can be consistent, however difference in noise and hysteresis will be observed. Noise is caused by random interference in practice, and hysteresis is caused by the difference between the actual system and the initial state of the simulation model.

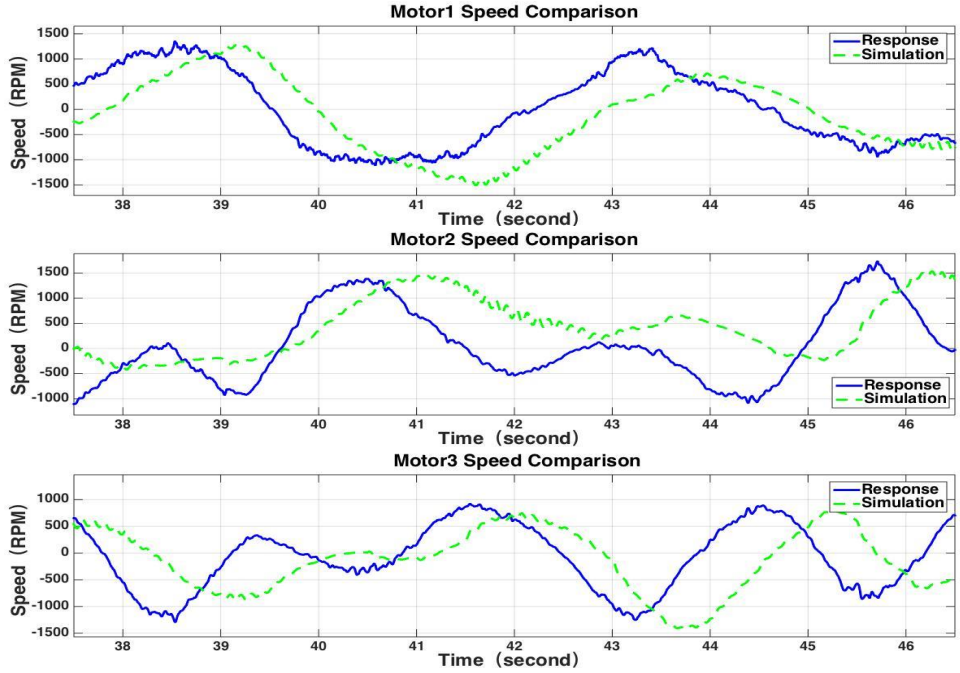


Fig. 15 Motor speed comparison

8. Conclusion

This work implements a control platform for Ballbot. We designed the controller and tested the performance of total system to optimize the simulation model and actual controller. The advantages of the control platform are shown experimentally. It is a powerful toolkit for the verification of the control method.

The merits of the control platform are:

(1) Graphical code generation allows engineers to quickly and easily change the algorithm without having to write a line of code.

(2) Engineers can modify the controller parameters in real time and observe the actual operation effect of the algorithm.

(3) Real-time operational data can be exported directly to facilitate follow-up study.

The significance of the control platform is:

(1) The seamless connection between MATLAB and the actual system ameliorates the difficulty of directly applying the controller designed for the simulation model to the actual system

(2) It provides an ideal platform for the application and optimization of the control method. The control engineer is freed from the constraints imposed by various engineering problems and can now focus solely on the development of the control algorithm.

(3) The unified software platform makes the design of the controller more efficient and quick by allowing direct debugging of the actual system, thus shortening the development cycle of the algorithm.

References

[1]. Information on: http://www.mathworks.com/hardware-support/beagleboard.html

[2]. Information on: https://groups.google.com/forum/#!topic/beagleboard/fC-Bg_-Ia34

[3].Informationon:http://www.adafruit.com/blog/2013/07/29/tutorial-introduction-to-the-beaglebone-black-device-tree

[4]. Information on: http://beagleboard.org/Support/bone101

[5]. Barret, F.S., Kridner, J. [2013]: Bad to the Bone: Crafting Electronic Systems with BeagleBone and BeagleBone-Black, Morgan & Claypool Publishers

[6]. Information on: http://www. mathworks.com/matlabcentral/fileexchange/39354-device-drivers

[7]. Liu Jie. Model Base Design. Beijing: Beijing University of Aeronautics and Astronautics Press,2011:1-6.

[8]. Zhang Yun-sheng. Design Principle and Application of Real - time Control System Software. Beijing: National Defense Industry Press,1998:1-12.

[9]. Liu Jun. Research on Electric Power Steering Control System Based on Rapid Control Prototype. He Fei: Hefei University of Technology,2009.12.14

[10]. Wu Hai-xiao, Fan Guang-qiang, Ma Cheng-guang. Development of CVT Electronic Control System Based on V-shaped Development Mode. Annual Proceedings of China Automotive Engineering Society. 2008: 585-589

[11]. NIKU, SAEED B. Introduction to Robotics: Analysis, System and Application. Beijing: Electronic Industry Press, 2004: 26-73.