Testing Robotic Systems: Case Study

Nikola Totev

Faculty of Mathematics and Informatics, Sofia University St. Kliment Ohridski,
5 James Bourchier Blvd., 1164 Sofia, Bulgaria

nikolart@uni-sofia.bg

Abstract. The field of robotics is experiencing rapid growth and is changing how we live our lives. Robotic systems are making their way into all areas of our life and by doing so present a number of challenges regarding safety. This is especially true in the fields of education and research where novel systems are being developed and tested. There are many articles focused on testing software systems in practice, but the same cannot be said about the testing of robotic systems. This paper aims to present such a study focused on the development and testing of a small balancing robot by analyzing the challenges of development, integration, and testing of the system. A series of steps that can be used to methodically test the components of a system and to verify that the desired functionality is implemented correctly are also proposed.

Keywords: Robotics, Control Systems, Testing Methods, Real-Time, Balancing Robot.

1 Introduction

The field of robotics is experiencing rapid growth and has the potential of revolutionizing how we live our lives. The introduction of low-cost sensors, actuators and other electrical components gives individuals, schools, universities, and research labs the opportunity to experiment more and to contribute to this field.

Companies such as Sparkfun, Adafruit, Pololu and many others provide plug-and-play components with powerful features, that users can easily incorporate into their projects. Even though most of these components are easy to use, to get the most out of them, a certain degree of technical knowledge is required.

A good example of this is sensor fusion and how it can be used to combine data from an accelerometer and gyroscope to calculate the inclination angle of the sensor.

Another example is the use of PID algorithms to control robots. This is part of the field of control theory, and it is an essential part of a variety of projects.

In most cases, the algorithms mentioned above are implemented from scratch and to determine if they work correctly, a testing approach needs to be considered.

This paper covers the challenges of developing and testing a small robotic system in practice and proposes a series of steps that can be used for testing other small robotic systems.

2 Testing in the field of Robotics

Robotic systems are heavily used in safety-critical domains such as healthcare, education, and transportation. This increase in interaction between the public and robotic systems drastically raises the chance of catastrophic failure [4]. This heightened risk makes testing of robotic systems a key part of the development process. It is imperative that Cyber-Physical Systems (CPS) [5-7], of which robotic systems may be considered a subcategory [4, 8] be thoroughly tested before being used in production environments.

Characteristics of robotic systems such as interaction with the physical world, and integration of hardware and software components, differentiate robotic systems from conventional software systems [4]. Due to these inherent differences testing approaches used for software cannot be directly applied to robotic systems.

The main differences between software systems and robotic systems are: (1) Robots are comprised of (unreliable and non-determinstic) hardware, software, and physical components [9-11]. This applies to both hobby grade and industrial grade products, but it is especially true when working with inexpensive components in a research or education setting. (2) Robots interact with the physical world via inherently noisy sensors and actuators and are sensitive to timing differences [11]. (3) Robots operate within the practically boundless state space of reality, making emergent behaviors (i.e., corner cases) difficult to predict [9]. (4) For robotic systems, the notion of correctness is often inexact and difficult to precisely specify [7].

These differences introduce numerous challenges related to the testing of robotic systems such as having to create heavy abstractions of physical reality or conduction real-world field testing [4] that can be time-consuming and expensive. In some cases, real-world testing may even be impossible because the environments that the system is aim towards are difficult to access. Examples of such environments are deep in the ocean, or on other planets.

There are a lot of studies focused on software testing practices, however none of them focus on the challenges in robotics [4]. To address this lack of research, a recent study interviewed a group of professionals working in the field of robotics [4]. The goal of this study was to identify common testing approaches, the challenges that are commonly encountered when developing tests for robots and the difficulties automating those tests.

The methods of testing mentioned by the participants are: (1) Field Testing, (2) Logging and playback, (3) Simulation Testing, (4) Plan-based testing and

(5) Compliance testing.[4] Out of these testing methods, simulation is one of the most seldom used, as simulators that accurately simulate the real world to a high degree just do not exist and are not suitable for testing of the whole system. Simulation is usually used as a high-level development tool [4].

The main challenges outlined in the study are: (1) Unpredictable corner cases, (2) Engineering complexity, (3) Culture of testing and (4) Coordination, collaboration, and documentation. [4]

The last thing the study discusses are the challenges of automating tests. As the previous paragraphs outline, testing on its own is difficult and trying to automate it makes things even more tricky. The main difficulties are: (1) Cost and resources, (2) Environmental complexity, (3) Distrust of simulation, (4) Software and hardware integration.

Due to all the outlined challenges, the area of robotic testing is still developing and further research into practical testing approaches is required.

3 Case study architecture

This section will briefly discuss the architecture of the system described in this case study – the Teensy Balance Bot. Robots of this type share some key characteristics, but there are many ways of implementing a balancing robot [3], because of this the specific architecture used for this case study is discussed.

3.1 Hardware

The Teensy Balance Bot, as the name suggests is based on the Teensy microcontroller family, in this case a Teensy 4.1 is used. The IMU module is the LSM6DS33 from Pololu, it has a gyroscope and accelerometer. The motor driver is the Cytron MDD3A and the motors are generic brushed motors with reduction gears. Two magnetic encoders from Pololu are used for detecting wheel rotation. A small OLED screen is displays status messages. The robot is powered by two Panasonic NCR18650PF Li-ion. The main body of the robot is custom made to fit the parts and manufactured using 3D printing.

The circuit diagram of the robot shows the connection between all the components (see Fig. 1).

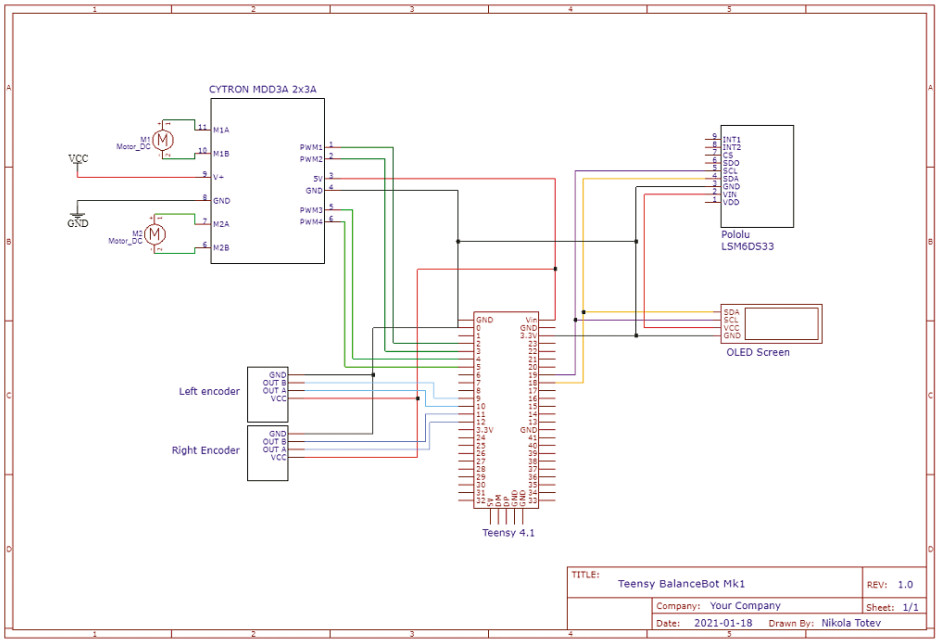

Fig. 1. Teensy Balance Bot circuit diagram.

3.3 Software

The software that runs on the Teensy microcontroller is written in C++ and uses a screen library from Adafruit. Everything else is built from scratch. Fig. 2 shows the complete architecture.

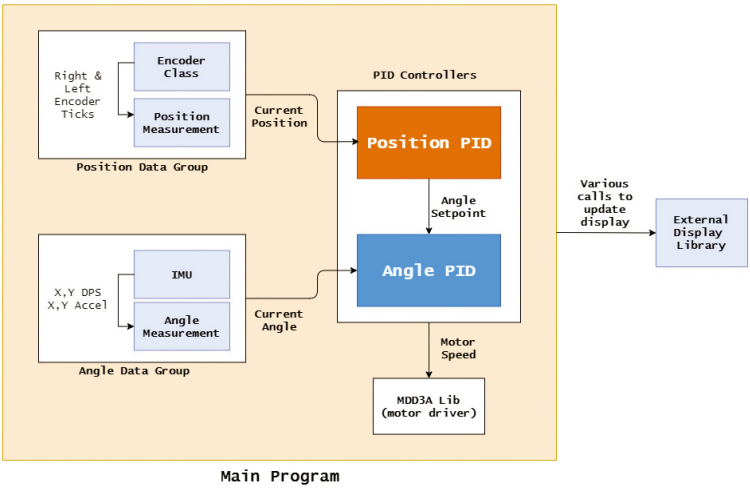

Fig. 2. Teensy Balance Bot Software architecture.

4 Proposed testing approach

Considering the challenges outlined in section 3, this section this section proposes a testing method that provides a systematic approach for testing robotic systems that fall into the category of balancing robots. Such robots are used in research and education and are an excellent way of learning how to work with sensors and actuators.

It is possible using this process to verify easily that the robot is working correctly, and that the desired functionality has been implemented.

4.1 Feasibility study

The testing phase of the project begins before any physical or software components are created. MATLAB and Simulink are a good option for simulating robotic systems.

Using a simulation allows for early detection of potential defects and in cases such as the balancing robot it helps to prove that the system can be created and controlled from a mathematical point of view. Simulations also allow for software to be tested before the hardware is available [4].

Using more advanced simulations and a more detailed model can also aid in the development of the software that runs on the microcontroller, however as section 3 outlined using simulation can be unreliable and refining the simulation can be expensive. In the case of the Teensy Balance Bot, a simulation was used to determine if the PID controller design was appropriate.

4.2 Testing tools

The next important step in this approach is choosing the right tools for testing. Since both hardware and software tests are being performed, appropriate equipment needs to be used.

Software testing tools

Software testing can be accomplished using the Visual Studio IDE with the VisualMicro plugin. This combination allows for debugging and provides an easy way to monitor the serial output from the microcontroller.

Hardware testing tools

An oscilloscope is used for verifying that the signals being sent from the sensors to the microcontroller or vice versa are correct. Oscilloscopes are an essential tool for testing as they are a window into the invisible world of electronics and troubleshooting a complex system like a robot without one, while possible, would be very difficult.

4.3 Divide and Conquer

Divide and Conquer is a popular strategy that is used in many fields. It is of particular interest in the testing of robotic systems because it allows for early problem detection and correction. Robots are made up of many components and diagnosing issues as a single system is difficult. Due to this, the approach proposed in this paper examines the case where each component goes through three stages:

Initial prototype – During this phase only basic functionality is developed and tested. Additional care is taken to not crowd the code with unnecessary functionality and the hardware is limited to the basic components. A good example of this is the development of the inclination angle measurement component. The first step in the development of this component is to validate that the IMU is function and producing data. The second step is to convert the raw data into a known unit and to verify that the values are correct. The third step is to implement a sensor fusion algorithm such as a Complementary or Kalman filter to measure an inclination angle [2].

Defining component I/O – In this step the inputs and outputs of the component are defined. This part embraces the black box model and allows for a good level of modularity. A key property of the approach proposed in this paper is modularity, even in the smallest projects, as it allows for much more flexibility.

Creating final version – During this phase, the component is implemented using best practices and conventions.

The three steps listed above are aimed towards software components that interact with hardware, such as sensors, as well as those that work exclusively with software components.

The “Divide and Conquer” approach falls into the category of Plan-based testing, outlined in section 3. It involves creating a rough plan and objectives for testing that can be specified in advance to manage and guide testing [4]. The participants in the study mentioned that they create system requirements list, which is used to ensure all components of the system are covered by the tests [4].

4.4 Final tests

Final tests of the system are the last step in this approach. In this phase, processes such as PID tuning are performed to dial in the behavior of the robot. PID tuning is done by varying the PID gains that are used in the algorithm, there are two main ways of accomplishing this. Before tuning a PID controller it is necessary to examine how each gain affects the behavior of the system.

As the name suggests, there are 3 gains to be tuned. “The proportional (P) action gives a change in the input (manipulated variable) directly proportional to the control error. The integral (I) action gives a change in the input proportional

to the integrated error, and its main purpose is to eliminate offset. The less commonly used derivative (D) action is used in some cases to speed up the response or to stabilize the system, and it gives a change in the input proportional to the derivative of the controlled variable. The overall controller output is the sum of the contributions from these three terms." [1].

The two methods of tuning a PID controller examined in this paper are using the system model together with built-in MATLAB functionality to calculate the gains and empirical testing.

Using results from the simulation. MATLAB has a useful feature that analyzes the model of the system and generates the appropriate PID gains automatically. This is a very quick and precise way to tune the system. The only downside to this method is that if the model that is being used is not accurate enough, the PID gains might not work well when used on the physical system.

Another approach is empirical testing. This is a more time-consuming approach, but it is easy for anyone to perform and it delivers good results. Empirical testing starts by setting the I and D gains to 0 and the P to 1. The P gain is then increased until the system oscillates, in the case of a balancing robot this means that the robot tilts back and forth. After that point, the P gain is decreased, and the D gain is increased. The D (derivative) gain has a damping effect that stabilizes the system. The last gain to be dialed in is the I gain; this affects the steady state error and should be implemented carefully to avoid integrator windup.

With the steps described above it is possible to easily tune a PID controller. This technique is suitable for small robots such as the Teensy Balance Bot or other projects used in research and education because during the tuning process the effects of each gain can be easily observed.

This testing approach falls into the category of "Field testing" outlined in section 3. In the same study, participants mentioned that during testing they relied on intuition to judge if something is working correctly or not. For the case of a small balancing robot, tuning the PID controller by testing it in the real world and using intuition to judge if the robot is performing correctly is a good way of efficiently testing the system.

More formal ways of testing such as using a simulation will always deliver better results, but as mentioned in section 3, simulations can be unreliable and might not replicate the real world correctly and that will lead to incorrect results. More time can be spent to refine the simulation, but for such a simple case as a balancing robot the cost of creating a detailed simulation is not justified.

5 Conclusion

The case study of a balancing robot discussed in this paper is a typical example of how a PID controller is designed, implemented, and tested for a real-world system.

The paper proposes a simple, but powerful modular approach to robot testing, that can be used in education or research settings. The presented approach gives insight into a practical implementation of the testing method by showcasing the testing steps of a balancing robot. Two PID testing approaches are discussed. In cases where field testing is an option an intuitive PID tuning approach that enables rapid testing and development of robots in research or education is outlined. In cases where testing the PID controller on a physical system is impractical, a simulation approach with both advantages and disadvantages is discussed.

6 Acknowledgments

This work has been accomplished with the financial support by the Grant № BG05M20P001-1.002-0011, financed by the Science and Education for Smart Growth

Operational Program (2014-2020) and co-financed by the European Union through the European structural and Investment funds.

References

1. Skogestad, Sigurd. (2001). Probably the best simple PID tuning rules in the world.
2. Hau-Shiue, Juang & Lum, Kai-Yew. (2013). Design and control of a two-wheel self-balancing robot using the arduino microcontroller board. 634-639. 10.1109/ICCA.2013.6565146.
3. J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas and T. Delbruck, "A pencil balancing robot using a pair of AER dynamic vision sensors," 2009 IEEE International Symposium on Circuits and Systems, Taipei, Taiwan, 2009, pp. 781-784, doi: 10.1109/ISCAS.2009.5117867.
4. A. Afzal, C. L. Goues, M. Hilton and C. S. Timperley, "A Study on Challenges of Testing Robotic Systems," 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), 2020, pp. 96-107, doi: 10.1109/ICST46399.2020.00020.
5. Duan, P. et al. "A Systematic Mapping Study on the Verification of Cyber-Physical Systems." IEEE Access 6 (2018): 59043-59064.
6. S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 9, pp. 1421–1434, 2016.
7. D. Marijan, A. Gotlieb, and M. K. Ahuja, "Challenges of testing machine learning based systems," in International Conference On Artificial Intelligence Testing, ser. AITest'19. IEEE, 2019, pp. 101–102.
8. S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," IEEE Systems Journal, vol. 9, no. 2,pp. 350–365, 2014.
9. L. Esterle and R. Grosu, "Cyber-physical systems: challenge of the 21st century," e & i Elektrotechnik und Informationstechnik, vol. 133, no. 7, pp. 299–303, 2016.
10. C. Hutchison, M. Zizyte, P. E. Lanigan, D. Guttendorf, M. Wagner, C. Le Goues, and P. Koopman, "Robustness testing of autonomy software," in International Conference on Software Engineering: Software
11. H. Li, Communications for control in cyber physical systems: theory, design and applications in smart grids. Morgan Kaufmann, 2016, ch. 1-Introduction to cyber physical systems.