Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2018, 5(3):498-502**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Firmware Test Automation using Open-Source Tools

Omkar Manohar Ghag

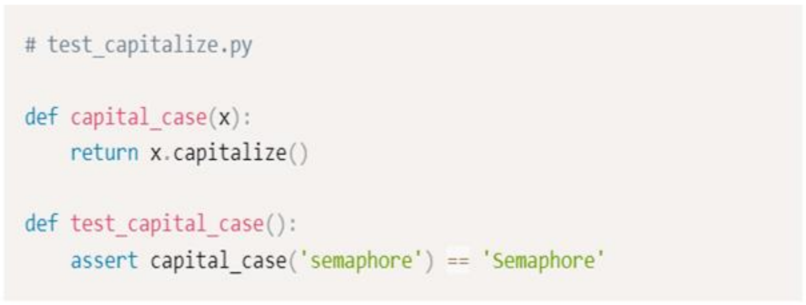
MS. In Telecommunication University of Pittsburgh, Pittsburgh, PA Independent Researcher

Abstract Python boasts a robust testing community and offers exceptional built-in testing capabilities within its standard library. The Python environment includes a wide range of testing tools. Pytest is notable for its user-friendly interface and ability to handle increasingly complex testing requirements. This paper presents a procedural method of how firmware testing can occur and how this testing promotes quality software. The open-source tool selected for this paper is Pytest, a Python-based tool. Python performs several functions in the automated testing procedures. The paper also presents an optimal process that is used to perform an optimization of the entire testing. It Prioritizes the task of writing the test cases. To ensure code quality and proper functionality, create test cases before implementation. Test-driven development is a widely used strategy in software development. Python offers several testing frameworks for utilization. It is both cost-free and openly available for modification. Git allows you to collaborate on projects of varying scales. By employing Git, you can modify your code and preserve or commit those changes at your discretion.
Additionally, this implies one can reverse any alterations they have already implemented. From the analysis, it is clear that the testing hardware has significant challenges when dealing with contemporary microprocessors, primarily due to their variety and intricate nature. Software-based self-testing is an alternative to hardware-based self-testing, where a microprocessor is tested based on its instruction set.

Keywords Pytest, Firmware, GitHub, Automated test pattern, NumPy.

1. Automated testing of firmware

In the present era, any proficient developer must possess the knowledge and skills required for program testing. Python boasts a robust testing community and offers exceptional built-in testing capabilities within its standard library. The Python environment includes a wide range of testing tools. Pytest is notable for its user-friendly interface and ability to handle increasingly complex testing requirements.
Pytest expects to find our tests in files with names starting with test_ or ending with _test.py. The file test_capitalize.py has a function named capital_case. This function takes a string as input and returns the exact string with all letters capitalized. In addition, we will generate a test called test_capital_case to validate the function's predicted performance. As pytest requires this specific syntax, we will add the prefix test_ to our function names.



```
# test_capitalize.py

def capital_case(x):
    return x.capitalize()

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'
```

Compared to the several assert something techniques in unit tests, pytest's utilization of a straightforward assert statement renders it significantly easier to comprehend and apply. Initiate the test by using the pytest command.

```
$ pytest
```

The provided code unambiguously demonstrates that the test has been successfully passed. Upon a meticulous examination of this code, it is evident that it does not validate the argument type as a string. An exception will be triggered if a number is provided as an argument to the function. One possible strategy for addressing this issue is to implement the function by throwing an exception with an informative error message.

```
# test_capitalize.py

import pytest

def test_capital_case():
    assert capital_case('semaphore') == 'Semaphore'

def test_raises_exception_on_non_string_arguments():
    with pytest.raises(TypeError):
        capital_case(9)
```

wepytest is an environment that offers extensive functionality and relies on plugins. It allows you to conduct thorough testing of your Python code. Pytest provides a variety of time-saving instructions and plugins that enable the execution of complex tasks while requiring less code for more straightforward tasks. It will automatically run all tests, including those conducted using the unit test framework upon initial setup. Although pytest's early development patterns may appear acceptable, they may cause aggravation as the testsuite grows. This issue is prevalent in the majority of frameworks. This session will demonstrate how to utilize specific capabilities of pytest to ensure the continued effectiveness and efficiency of testing as it expands.

2. Test Version control

After conducting the firmware testing with Pytest, it is now necessary to establish the test controls. In essence, Git is utilized to carry out version control. Git is a decentralized version control system that enables us to monitor file modifications. It is both cost-free and openly available for modification. Git allows you to collaborate on projects of varying scales. By employing Git, you can modify your code and preserve or commit those changes at your discretion. Additionally, this implies that the user can reverse any already implemented alterations.

3. What is GitHub?

GitHub, a web interface, provides a convenient method for storing and managing Git repositories. It facilitates collaboration among several developers by enabling them to share the code for a single project. While other developers are engaged in the project, we can also incorporate modifications to it.

4. Guide for Configuring Git: A Primer for Novices.

Use this command to assign a user's login, email, and branch, which enables tracking of the author of a commit made to a project. Once Git is installed on the computer, it can be customized according to the requirements using this command. To demonstrate, execute the below commands in the terminal: git config --global user. The name "[username]" and git config global user.email [email address].

5. A Comprehensive Guide to Git Init for Beginners

To commence Git within your project, you might utilize the gitinit command. To initiate Git for a project and begin tracking changes, use this command to initialize Git.

A Comprehensive Guide to the Git Add Command for Beginners.

Executing this command will incorporate the file into the staging area. The files that experience modifications are placed in the staging area, awaiting the subsequent commit. The Git add command is employed to include files in the staging area. All contents within the folder have been transferred to the staging area. Designate a specific file for staging; use the "git add (file name)" command. To persist edits to a file and implement them in the project, utilize this command—a comprehensive guide for beginners on Git Commit.

All files present in the staging area, as well as any files that have been added using the Git add command, will be committed simultaneously. For instance, This is the initial commit executed using Git. The Git repository is permanently stored using this command. If the user utilizes the Git add command to include a file in the staging area, this information is required.

Establish a testing facility with the requisite hardware devices to conduct software testing. Hardware testing includes verifying its proper functionality, stability, and absence of any procedural flaws[1]. It also incorporates tests that significantly strain the CPU and memory to assess performance and durability.

The complexity of contemporary hardware design necessitates using testing methodologies that can effectively address emerging issues. Consequently, the advancement of test development with new hardware standards is happening quickly.

Hardware testing encompasses various components, including the BIOS and CPU/Processor. Conduct examinations on the hardware to ensure its adherence to all regulations and logical integrity. Functional tests are commonly used in hardware testing to determine if a product has met the test requirements.

Automated test pattern generation (ATPG), built-in self-testing (BIST), and software-based self-testing are all instances of software-based testing. The testing hardware has significant challenges when dealing with contemporary microprocessors, primarily due to their variety and intricate nature. Software-based self-testing is an alternative to hardware-based self-testing, where a microprocessor is tested based on its instruction set. Software-based self-test offers the benefit of being applicable throughout the microprocessor's regular operation mode, enabling swift implementation of the required tests.

The Automated Test Pattern Generator (ATPG) is a software tool that automatically generates test patterns for testing electronic circuits. The most straightforward and uncomplicated method does not necessitate a thorough comprehension of the DUT (Device under test). It commences with a chip netlist, incorporates scan chains (a testing design technique), and produces vectors. Modern EDA (Electronic et al.) tools utilize scan chains to determine the optimal division of a design into blocks.

6. Jira Software for bug tracking

Jira Software monitors and manages every stage of the software development lifecycle, encompassing the efficient recording, allocation, and ranking of defects. Jira's powerful workflow engine provides a clear and precise view of the current status of an issue and promptly notifies when problems transition from the backlog to the done state. Jira Software facilitates seamless collaboration among the software team, enabling users to monitor every product development stage with complete transparency. Due to the restricted availability of internal networks resulting from intricate designs, many methods have been developed to enhance the testability of systems. One such method is the implementation of Built-In Self-Test (BIST), which is particularly suitable for testing intricate systems. The execution of BIST relies on a foundational design known as a complete scan architecture. This signifies that the storage components of the DUT combined to form several scan chains. This method allows test patterns to be transferred into and out of the storage elements sequentially. An extensive and expensive external testing system is not required for Built-In Self-Test (BIST) to operate. All the necessary testing has already been completed; users only need a tester to initiate the process.

Firmware Testing

Engineered products and systems require firmware, a computer program integrated into hardware, to perform tasks such as operating, monitoring, and manipulating data. The device's functionality is governed by the firmware, which operates at a low level. Firmware is present in various electronic devices, including embedded systems, desktops, laptops, servers, and mobile phones[1]. Firmware testing is essential since it ensures that the firmware system meets all standards in terms of functionality, performance, operations, and implementation. The subsequent action involves optimizing efficiency while minimizing potential hazards.

7. Device Farm

Achieve superior performance and functionality for online and mobile applications by leveraging AWS Device Farm, a cutting-edge application testing service. It enables firmware testing on various desktop browsers and real mobile devices without the requirement to deploy or manage testing equipment. To enhance your test suite's speed, you can use this service to execute it on multiple desktop browsers or physical devices concurrently. Additionally, it generates logs and videos to speed up identifying faults in an application.

8. There are three techniques available for evaluating firmware.

Examines the code of complete microprograms or examines the machine states following execution to conduct testing at the microprogram level. Secondly, it assesses the extent to which individual microinstructions are considered by analyzing the machine states after execution or investigating the allocation of micro-operations. Monitors the execution of individual micro-operations to ensure their accuracy and adherence to specified criteria. Although the process of testing firmware is extensive and intricate, automated solutions can assist in its completion[1]. FWTS, often known as the Firmware Test Suite, is a Linux software designed to automate testing firmware. These tests assess several aspects of a computer's firmware, including ACPI, UEFI, hardware configuration, power management, and other functionalities.

9. Creating cron jobs using Jenkins.

Jenkins is a popular software tool extensively utilized for continuous integration and delivery. The strength is in the comprehensive array of plugins that enable the modification of virtually every aspect. A dynamic community has created a plethora of plugins. We automated various processes on a highly utilized e-commerce platform using Linux crontab every midnight. Many of these tasks are interdependent. Due to its many beneficial features, the declarative pipeline is a highly favored plugin for Jenkins.

A pipeline step will generally only advance if the preceding phase has completed its processing. Unlike crontab, this allows the users to schedule subsequent jobs without being restricted to a set time. Their execution will be delayed until the completion of the preceding task. When a job fails, users may either retry it or go with the next item in the pipeline.

While pipeline stages are commonly linked to deployment operations such as code creation and test execution, they possess various applications and can be utilized for different activities. Jenkins maintains a record of the chronological sequence of pipeline runs and the log of each step's execution. This enables easy referencing and troubleshooting at any given moment. By utilizing Jenkins' SSH plugin, I can manage all of my tasks from a distinct server, allowing the users to execute commands remotely. This relieves the users from managing crontab on each target server and helps them utilize a dedicated task management server to run jobs remotely. The time and visual depiction of each step's execution are displayed in the pipeline. The dashboard, depicted below, facilitates the monitoring of job statuses.

Conclusion

Pytest is an excellent choice if the users need a straightforward, adaptable, open-source test automation framework based on Python. Pytest is highly favored by testers for its prominent capability to test APIs, but it also possesses the capacity to conduct tests for databases and user interfaces. It can function independently for testing purposes or be incorporated into Python web frameworks like Flask or Django to enhance the dependability and functionality of its unit tests. In Pytest, tests can be executed as a whole or selectively using tags or names. Tests in Pytest are Python functions. To utilize Pytest's built-in parallel test running feature, provide an extra option in the command line interface and execute the pytest run command.

References

[1]. Costin, A., Zarras, A., & Francillon, A. (2016, May). Automated dynamic firmware analysis at scale: a case study on embedded web interfaces. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (pp. 437-448).

[2]. Ahn, S., & Malik, S. (2014, October). Automated firmware testing using firmware-hardware interaction patterns. In Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis (pp. 1-10).

[3]. Vilajosana, X., Tuset, P., Watteyne, T., & Pister, K. (2015). OpenMote: Open-source prototyping platform for the industrial IoT. In Ad Hoc Networks: 7th International Conference, AdHocHets 2015, San Remo, Italy, September 1-2, 2015. Proceedings 7 (pp. 211-222). Springer International Publishing.