



## From Design to Test Reliability: Inside the Engineering of PaytestRobot

---

Automation in payment software QA is usually discussed in terms of software - test frameworks, scripts, and pipelines. But when physical payment terminals are involved, physical interaction becomes just as important. Buttons must be pressed, cards must be inserted, and interfaces must be navigated exactly as a user would do it. If those interactions are inconsistent, test results can quickly become unreliable.

This is where mechanical design plays a key role. PaytestRobot was built to combine controlled mechanical movement with software-driven test execution, allowing QA teams to repeat the same interactions again and again in exactly the same way. When the same motion and sequence of actions are executed consistently, issues that might be missed during manual testing can appear in a predictable and reproducible way, making them easier to detect and resolve.

### Why Reliability Problems Appear in the Physical Payment terminal QA Processes

QA teams usually face reliability problems when the number of configurations they need to test begins to grow. Payment software often runs across different terminals, different versions, and different customer setups. Each of these setups may enable or disable certain features, such as DCC, specific card schemes, reservation flows, or hospitality features like tipping.

Over time, ensuring every feature works correctly across all of these combinations becomes impossible with manual effort alone. Testing cycles take longer, and teams may run out of time to validate every scenario before a release. When that happens, some configurations receive less attention and reliability issues can start to appear.

Manual testing also introduces another challenge. When testers repeat the same flows many times, they begin to recognize patterns and may assume the results will be the same each time. If several transactions work correctly in a row, it becomes easy to trust the outcome and stop checking every detail as carefully as before. A receipt might not be reviewed closely, or a final step might be skipped because everything seemed to work previously. Over time, this repetition can lead to missed issues and reduce the reliability of the testing process.

## The Design Behind PaytestRobot

The idea for PaytestRobot originally came from challenges in payment software development. At the time, releases often happened only every few months. Bugs could remain hidden in the system until the software finally reached QA testing. When problems were discovered that late in the process, fixing them was often complicated because additional features had already been built on top of the faulty behavior.

The goal of PaytestRobot was to bring testing earlier into the development cycle. The first goal was simple: run basic payment transactions automatically every morning to confirm that the core flows still worked. If something broke during development, the team could detect it immediately instead of discovering the issue weeks later during a full QA cycle.

From the beginning, several design principles were important. One was flexibility. The PaytestRobot needed to support different types of terminals rather than being built for a single device. Instead of creating a custom PaytestRobot for each terminal, the design uses interchangeable base plates so different devices can be mounted on the same PaytestRobot. Another key requirement was that the PaytestRobot should perform the same actions as a human tester - pressing buttons, inserting cards, and interacting with the terminal interface in a predictable and repeatable way.

## Motion System - Where Precision Begins

The motion system of PaytestRobot is based on a Cartesian coordinate model using X and Y movement. This design was influenced by early work with 3D printers. One of the engineers involved in the project had recently built a 3D printer, which uses the same coordinate-based movement system, and that concept was later applied to the PaytestRobot.

With this approach, our PaytestRobot moves along defined axes to reach a specific position before performing an action such as pressing a button. This makes it possible to position the PaytestRobot in exactly the same place every time. That level of consistency is vital for terminal test automation because some software bus only trigger after hundreds of identical transactions.

When a test case requires the same sequence of actions, the PaytestRobot can reproduce those movements consistently. If a bug depends on a specific interaction pattern, repeating that pattern makes the issue easier to detect and analyze.

To maintain positioning accuracy during long test runs, the PaytestRobot uses stepper motors. These motors move in precise steps, and each step represents a fixed amount of movement. The electronics track these steps so the system always knows the PaytestRobot's position. Periodically, the PaytestRobot performs a homing process, returning to a reference point before continuing execution, ensuring that concurrent terminal control remains accurate even after thousands of testing cycles.

## Actuation and Physical Interaction

Designing the mechanism that physically presses on a terminal interface was one of the more challenging parts of building the PaytestRobot. The system used in PaytestRobot went through several iterations as the team improved how the PaytestRobot interacts with different devices.

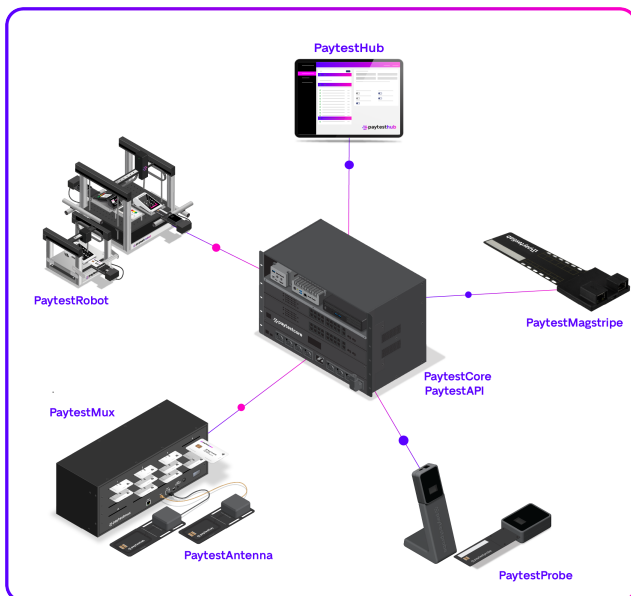
Early versions relied on simple servo motors, but these components eventually wore down under repeated use. As the design evolved, stronger components and improved control mechanisms were introduced to make the system more reliable over long testing cycles.

Touchscreen interfaces added another challenge. With touchscreens, the timing and speed of a press are important. If the PaytestRobot presses too slowly, the device may interpret the action as a double tap. If it releases too quickly, the tap may not register at all. To address this, the pressing mechanism was refined to control both the force and timing of each interaction.

Different devices also require different press strengths. For touchscreens or mobile devices, the robot uses a soft press of around 300 grams. Other devices, such as vending machines or more rugged terminals, require stronger interaction. In those cases, the PaytestRobot can apply between 600 and 800 grams of force depending on the device.

Terminal layouts can introduce additional mechanical challenges as well. Some devices have motorized card readers or place components very close together, leaving limited space for both card insertion and button interaction. These variations require careful mechanical adjustments when building automation setups.

## Hardware and Software Coordination



Reliable automated payment testing depends on close coordination between the mechanical system and the software that controls it. Both must work together so that every physical interaction happens at the correct moment during a transaction.

PaytestRobot connects to a PC through a network interface, and commands are sent using G-code, a language commonly used to control CNC machines and 3D printers. These commands determine where the PaytestRobot moves and what actions it performs.

Within the cloud-based payment testing software, each test case defines a sequence of steps. A transaction might require inserting a card, waiting for the PIN screen to appear, entering the PIN, and then ejecting the card. If those actions occur in the wrong order, the transaction will fail. Because of this, proper synchronization between the test logic and the PaytestRobot's movements are essential.

Movement and actuation are controlled by an electronics board called the Smoothieboard. This board powers the motors and translates commands from the PC into the physical movements needed to run the test.

To make sure the PaytestRobot interacts with the correct elements on the terminal, the system uses layout files that map coordinates to specific positions on the interface. Older terminals with simple keypads were easy to map. Modern terminals often have more complex interfaces, so cameras can be used to observe the screen and determine where elements appear. The PaytestRobot can then move to those coordinates and perform the required interaction.

## Real Lab Impact and Future Challenges

Automated testing with PaytestRobot allows QA teams to run the same transaction far more times than would be practical with manual testing. A human tester might repeat a transaction several times, but repeating it hundreds or thousands of times quickly becomes unrealistic. The PaytestRobot, however, can execute the same flow continuously without interruption.

Running repeated transactions makes it easier to observe how terminal software behaves over time. For example, teams can detect whether memory usage gradually increases or whether the terminal eventually stops working after extended operation. Issues like these often appear only when the same transaction is executed many times in a row.

Automation also allows teams to test a broader range of transaction types and configurations. When test coverage is well defined, these scenarios can be executed repeatedly to confirm that the system continues to behave correctly. This provides faster feedback when something breaks and allows testers to spend more time on exploratory testing rather than repeating routine flows.

As testing environments scale, new technical challenges appear. When several terminals are placed close together, contactless signals can sometimes interfere with each other, causing a terminal to detect a signal intended for another device. Larger setups can also require multiple cameras and OCR systems to monitor different terminals at the same time. Managing these systems may require more powerful PCs, better bandwidth management, and careful hardware configuration.

## Continuous Improvement

Even with its current capabilities, PaytestRobot continues to evolve. One area of improvement is cable management, helping keep test environments organized and easier to maintain. Another focus is refining the card insertion mechanism so that cards can be inserted consistently and reliably during automated testing.

Improving calibration and consistency in card insertion remains an important engineering challenge, and development continues to focus on making this interaction as stable as possible across different devices.

As payment systems evolve and testing environments grow more complex, these ongoing improvements help ensure that PaytestRobot remains a reliable tool for automated payment testing.

Matija Mazalin  
Head of Technology



sales@paytestlab.com  
PaytestLab  
Förlibuckstrasse 66  
8005 Zürich, Switzerland

[www.paytestlab.com](http://www.paytestlab.com)

