Deploying a Low-Cost Vision-Guided Cartesian Robot for Physical Interaction in Mobile Testing: An Industrial Case Study

Matheus Lopes
Motorola Mobility
Brazil
matheusl@motorola.com

Breno Miranda
Federal University of Pernambuco
Recife, Brazil
bafm@cin.ufpe.br

ABSTRACT

Automation of tests and related tasks is becoming increasingly essential, as it significantly reduces manual effort and improves efficiency. However, certain test cases cannot be fully automated due to the need for physical interaction either during or prior to their execution. In mobile testing scenarios, for instance, physical interaction may be required to enable USB debugging when it is not pre-authorized, preventing the full automation of test suites. Accordingly, the use of robotic mechanisms for software testing has emerged as a viable alternative to overcome the limitations of full automation. However, such solutions often involve high acquisition, development, and integration costs. To address these challenges, this paper proposes an alternative low-cost approach: a Cartesian robot controlled via Arduino, equipped with a 3D-printed finger. The robot's movements and interactions with the device screen are guided by computer vision, enabling automated physical interaction during testing. This setup enables automated testing while significantly reducing development and operational costs compared to conventional robotic arms typically used for such tasks. As a preliminary evaluation, our proposed solution was deployed in an industrial setting to enable USB debugging and unblock the automation process within a software testing framework. The results indicate that the approach is highly accurate. Furthermore, when compared to the manual process, which takes approximately 120 minutes, our method reduces the time to 70 minutes, resulting in a time savings of approximately 42%.

KEYWORDS

Software test automation, robotics, computer vision, USB debugging

1 Introduction

In the context of mobile software development and testing, there is a growing demand for higher levels of test coverage, agility, and reproducibility, particularly within continuous integration and continuous delivery (CI/CD) pipelines [8, 12]. However, manual testing poses significant challenges in meeting these demands. It often lacks reproducibility and suffers from limited scalability due to its labor-intensive nature.

Although manual testing remains essential for identifying specific issues, such as edge cases, usability flaws, and user experience problems, it continues to rely heavily on physical interactions, which further constrains automation and repeatability. While solutions like robotic arms can help overcome these limitations, they often involve high acquisition, maintenance, development, and integration costs, making them a less attractive or deprioritized option in many development environments. As noted by Afzal [1], "Integration and system testing are mostly performed manually in the field. While field testing is an important part of robotics development, it can lead to expensive and dangerous failures. Furthermore, field tests are limited by the scale of scenarios and environments that can be simulated."

Hardware validation scenarios involving movement dependency and tactile interaction with the screen require physical interactions that standard software frameworks, such as Appium [4] and UIAutomator [3], cannot fully simulate. Semi-automation is often used as an alternative, but human-in-the-loop dependencies reduce test efficiency and increase cycle times. As Afzal [1] notes, *"The need for a human-in-the-loop makes it difficult to execute tests frequently, and automation is limited to specific tasks or components"*.

Therefore, in light of these challenges, this work proposes a low-cost, modular automation solution based on the Arduino control system [5], which is widely available on the market, along with 3D-printed components. By integrating actuators and servomotors into a customizable workbench setup, the system enables the emulation of simple to complex user interactions, achieving full automation. Furthermore, with support for serial communication, it integrates seamlessly with widely adopted testing frameworks, enabling CI/CD workflows without requiring any manual steps. The main contributions of this work are:

- An analysis of the primary limitations to achieving full automation in mobile software testing, including a complexity comparison with existing solutions.
- A detailed description of the design and implementation of a low-cost, modular automation system based on Arduino and 3D-printed components, enabling tactile interaction with smartphone devices.
- An evaluation of the proposed solution in an automation-blocking scenario. Specifically, enabling USB Debugging via screen touch, achieving over 86% accuracy.

The remainder of this paper is organized as follows. Section 2 presents the context and motivation for this research. Section 3 provides background information, including a brief overview of Cartesian movement. Section 4 describes the proposed solution, the Cartesian Robot. Section 5 details the evaluation conducted and discusses the results. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Context and Motivation

Mobile Automation Challenges. Automating mobile test cases presents several challenges, particularly due to the need for sensory feedback, visual cues, or physical interaction with the device. These challenges are especially evident in test cycles that require the Android Debug Bridge (ADB) [2], as the automation is constrained by the steps needed before and after enabling this feature.

- **Sensory Feedback:** Test steps that rely on physical responses from the device, such as vibration, haptic feedback, or sounds.
- **Visual Feedback:** Test steps that require verifying the device's visual responses, like screen brightness levels or the on/off status of the display.
- **Allow Debugging Features:** Test scenarios that require enabling USB debugging to perform commands, inputs, and intents, facilitating the execution of automated test scripts. This process necessitates physical interaction, as shown in Figure 1a.

(a) Pop-up (b) Targets (c) Detection

Figure 1: USB Debugging pop-ups

Physical Interaction. Certain test cycles may be blocked by elements that require physical interaction with the device, such as pressing a hardware button, plugging in a cable, or performing a tactile action that cannot be simulated by software. These elements make the test non-automatable, preventing the full automation of the process, as they cannot be replicated by commonly used frameworks like Appium or UIAutomator. In such cases, hybrid solutions, often involving integration with physical robots or manual intervention [10], are necessary. However, human involvement compromises scalability and limits the ability to run tests continuously without supervision. While robotic solutions can mitigate this issue, their integration and acquisition often present significant challenges, acting as a critical barrier to widespread adoption.

3 Background

Movements performed within an orthogonal coordinate system, defined by the X and Y axes, are called Cartesian movements. Cartesian systems utilize actuators that move an object or tool linearly and precisely along each axis.

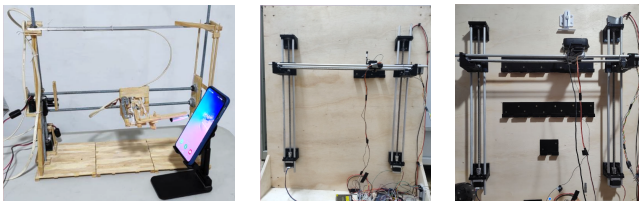
This type of system is widely used in 3D printers and CNC machines due to its easy control and trajectory computation. Cartesian motion enables highly precise movements, making it advantageous in physical test automation scenarios where the device under test (DUT) must receive manual inputs or tactile interactions.

Cartesian mechanisms controlled by microcontrollers can, for instance, simulate the human gesture of touching a smartphone screen during a test or physically interact with hardware. Furthermore, these systems enable full automation when integrated with testing, automated control, verification, and feedback frameworks.

4 The Proposed Solution

The Cartesian Robot was developed to address the constraints on complete test automation, due to the need for physical interaction, while also minimizing development, implementation, and integration costs. By utilizing a simple Arduino-based control system and Cartesian motion, it facilitates ease of integration, replicability, and rapid development. Moreover, the design of the system allows a flexible actuation area and a highly customizable architecture.

Implementation. The implementation of the proposed solution began with prototyping several robot models to identify the optimal configuration for performing two-axis movements and tapping the device screen. As a proof of concept, Prototype 1 (Figure 2a) was developed. Prototypes 2 and 3 (Figures 2b and 2c) were initially designed using 3D CAD software and subsequently constructed at full scale. They were built using a combination of plywood blocks, 3D-printed components, stepper motors with drivers, a servo motor, limit switches, a 3D-printed finger made from conductive filament (Figure 3a), and an Arduino Mega controller.



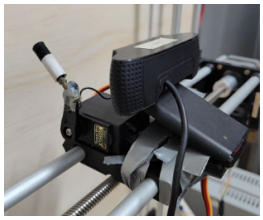
(a) Prototype 1 (b) Prototype 2 (c) Prototype 3

Figure 2: Three prototypes of the Cartesian Robot

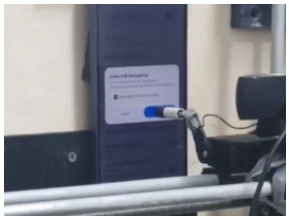
Computer Vision Feedback. The first version of prototype 3 (Figure 2c) was limited to two-axis (X, Z) movements that were hardcoded at the software level. This limitation reduced positional accuracy, as smartphone models vary significantly in screen dimensions. To address this issue and enhance both accuracy and reliability, a webcam was integrated into the movement head to provide visual feedback of the movements. This feedback not only guided subsequent actions but also contributed to the overall performance improvement of the system. To further refine the system, an object-detection model based on the YOLO architecture, initially proposed by Redmon et al. [11], was employed. We utilized YOLOv8 [13], which was specifically trained to recognize key objects and artifacts on the device screen (Figures 1b and 1c).

Scripts Communication. In the context of the scripts-actuator architecture, the code uploaded to the Arduino receives commands via UART serial communication [6], which enables integration with various programming languages due to its low-level communication layer. After receiving a command, the Arduino decodes it, the robot executes the corresponding action, and then returns a confirmation of execution via serial communication. Movement sensing and direction control utilize image recognition through YOLO [13]. This module is implemented in Python, which processes the captured frame to recognize the target, calculates the pixel distance between the frame's capture center and the detected object, converts this information into a control command, and transmits it to the Arduino via serial communication using pySerial [9]. Once the object is centered, a command to simulate a touchscreen press is sent to the Arduino (Figure 3b).

Final Proposal. The final version features six smartphone slots and can interact with each device across the entire touchscreen area. After power-up, during idle periods or for homing and calibration, the robot returns to its home position. It then executes movements according to the received commands and, at the end, performs the tap on the screen (Figure 3b).



(a) 3D printed finger



(b) Touching USB Debugging

Figure 3: WebCam and Finger

5 Evaluation

To evaluate the proposed solution, we selected a process that typically precedes and interrupts automated testing cycles: enabling USB Debugging via screen touch. This task presents several challenges, as the device screen contains multiple interactive elements. Moreover, variations in screen size, resolution across different devices, as well as low-light conditions and screen reflections, further complicate reliable automation.

For the evaluation, 10 operations were performed on each of the three selected device models, with screen resolutions of 1080×2400, 720×1612, and 1220×2712 pixels. Tests were carried out under real-world conditions, taking into account variables like luminosity, dark or light themes, and other factors. Metrics analyzed included travel time, object recognition, centralization, total interaction, and accuracy.

Table 1 presents the results of our evaluation. We observed high accuracy in the interactions, with an average 86.66%, along with lower deviations in metrics such as time to centralize, travel, and target recognition. When compared to manual execution, our proposed automated solution reduces the execution time from 120 minutes to 70 minutes, achieving a 42% reduction. Given that this cycle is performed at least twice a week, our approach saves over 86 hours annually, equivalent to nearly 11 full days of a human tester's time, which could be redirected toward other tasks.

Table 1: Test Results

Model	Centering	Travel	Iteration	Accuracy
Model A	11s	20s	2	100%
Model B	10s	20s	1	70%
Model C	12s	20s	2	90%

6 Conclusion and Future Work

This paper explores the use of Cartesian movement and computer vision in the context of mobile testing automation, focusing on increasing the number of automated test cases alongside a low-cost implementation and development strategy. The proposed Cartesian Robot aims to reduce testing time and the manual effort associated with non-automatable scenarios by enabling physical interaction with the device in a controlled and repeatable manner.

By reducing reliance on human intervention, the system contributes to improved repeatability, scalability, and reduced delivery times for mobile application testing. This aligns with observations in prior work, such as Afzal [1], which highlights the predominance of manual testing in integration and system-level scenarios.

Besides that, this solution can be more attractive when considering the costs compared to common industrial robotic arms. While a typical 6-axis robotic arm ranges from R$15.000,00 to over R$300.000,00 depending on the manufacturer, payload, and precision, the proposed Cartesian solution can be built for under R$2.000,00 and has less complexity.

In addition, the Cartesian solution has a flexible actuation range that can be easily adapted to different sizes or movement profiles allowing for an increased number of devices in parallel, while industrial robotic arms, although more advanced in terms of movement accuracy, are often constrained by their joint geometry and kinematics, especially in confined testing environments.

Although the solution does not yet fully eliminate all manual procedures, it demonstrates that affordable robotic systems can play a key role in bridging the gap between fully manual and fully automated testing. Future work includes expanding the range of physical actions, integrating with existing test frameworks such as Appium, and evaluating the system in continuous integration (CI) environments, also using CoreXY [7] as a new way of movement.

ACKNOWLEDGMENTS

This work was supported by the research cooperation project between Motorola Mobility Comércio de Produtos Eletrônicos Ltda (a Lenovo Company) and CIn-UFPE, and by a grant from the National Council for Scientific and Technological Development (Grant CNPq-Universal 408651/2023-7).

REFERENCES

[1] Afsoon Afzal. 2021. *Automated Testing of Robotic and Cyberphysical Systems.* Ph. D. Dissertation. Carnegie Mellon Univeristy. https://afsafzal.github.io/materials/thesis.pdf
[2] Android Developers. 2025. Enable USB Debugging. https://developer.android.com/studio/debug/dev-options#enable Accessed: 2025-06-03.
[3] Android Developers. 2025. UI Automator. https://developer.android.com/training/testing/ui-automator. Accessed: 2025-06-03.
[4] Appium Contributors. 2025. Appium: Mobile App Automation Made Awesome. https://appium.io. Accessed: 2025-06-03.
[5] Arduino LLC. 2025. Arduino — Open-source electronics platform. https://www.arduino.cc. Accessed: 2025-06-03.
[6] Jan Axelson. 2007. *Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems.* Lakeview Research.
[7] CoreXY Project. 2025. CoreXY: A New Foundation for Motion Systems. https://corexy.com. Accessed: 2025-06-03.
[8] GitLab Inc. 2024. What is CI/CD? https://about.gitlab.com/topics/ci-cd/. Accessed: 2025-06-03.
[9] Chris Liechti. 2025. pySerial - Python Serial Port Extension. https://pyserial.readthedocs.io/ Accessed: 2025-06-03.
[10] Lucas Maciel, Alice Oliveira, Riei Rodrigues, Williams Santiago, Andresa Silva, Gustavo Carvalho, and Breno Miranda. 2022. A systematic mapping study on robotic testing of mobile devices. In *2022 48th Euromicro conference on software engineering and advanced applications (SEAA).* IEEE, 475–482.
[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*
[12] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. 2017. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE access* 5 (2017), 3909–3943.
[13] Ultralytics. 2023. YOLOv8: Cutting-Edge Object Detection Models. https://github.com/ultralytics/ultralytics. Accessed: 2025-06-03.