

Appendix 2: Testing Operations Experience and Reflections

Conceiving the Testing Process and Study Design

The primary purpose of the testing programme was to obtain focused, actionable insight into the Blackbox prototype across hardware, software, and user experience dimensions. The intention was not to validate market readiness, but to identify which functions were viable, which required improvement, and where the most impactful iteration efforts should be concentrated in order to move the product closer to a market-ready state. The testing was therefore positioned as a directional and diagnostic exercise.

From the outset, the study was deliberately designed to be lightweight. This decision was driven by cost considerations, the early-stage nature of the hardware, and the desire to move quickly without over-engineering the testing framework. Rather than seeking broad coverage, the process leaned heavily on the expertise of a small number of experienced testers who were already familiar with relevant domains such as hardware, software systems, or user experience. This approach prioritised depth and quality of feedback over number of testers.

A core design principle in the testing conception was to think explicitly about the end users of the system. This included not only the merchant operating the device, but also the buyer or retail customer interacting with it during a transaction. Questions of convenience, clarity, and seamlessness on both sides of the transaction were deliberately incorporated into the testing prompts, ensuring that feedback was grounded with the end-user in mind.

Designing the Testing Instructions and Feedback

The testing instructions and feedback form were developed through a combination of informed research and consultation, with the aim of identifying the key factors that should be examined for a prototype of this nature. The resulting pack was intended to serve as a comprehensive point of reference for testers, guiding them through specific functions while also prompting broader reflection on usability, clarity, and workflow.

To anchor the feedback in established theory, the study integrated human-computer interaction principles, specifically drawing on Jakob Nielsen's usability heuristics. These heuristics provided a structured lens through which testers could assess the system, helping to frame observations around system status visibility, error handling, consistency, and efficiency. This also ensured that feedback could be interpreted systematically rather than remaining purely anecdotal.

The feedback form was intentionally designed to capture both qualitative and quantitative input. Testers were encouraged to provide descriptive reflections, suggestions, and observations alongside numerical ratings, with the aim of balancing richness with comparability.

Limitations of Study Design

In retrospect, the numerical ratings introduced a degree of subjectivity that limited their analytical value. Testers interpreted the rating scale differently: some viewed a “5” as meaning “acceptable with no issues,” while others interpreted it as “excellent and market-ready.” This form of perception bias or scale-use heterogeneity made it difficult to draw meaningful comparisons or aggregate conclusions from the quantitative data alone. Greater standardisation—either through clearer definitions or a different scale design—would likely have reduced this ambiguity.

Another limitation stemmed from the deliberately small sample size. While this kept prototype production costs low, it constrained the ability to perform meaningful statistical analysis or significance testing. In hindsight, an alternative design—such as region-based shared testing, where multiple testers rotate through fewer devices—could have increased the sample size without increasing hardware costs. This would have improved the robustness of quantitative insights.

That said, it is also important to recognise that the product was still a prototype undergoing significant change. A much broader or more statistically rigorous testing programme may not have been appropriate at this stage. Wider field testing is likely more suitable once the product is closer to market readiness, where stability is higher and iteration cycles are slower.

Execution of the Processes

Recruiting and Organising of Testers

Given the lightweight design of the study and the limited number of available devices, tester recruitment focused on experience and relevance rather than representativeness. Testers were recruited through ecosystem contacts, teams with relevant expertise such as Cryptape and Nervape, and personal networks. This ensured a high signal-to-noise ratio in the feedback received. Testers were given a link to the testing instructions and feedback back, and advised to create their own copy which they could fill in and submit.

In retrospect, this was a reasonable way to collect feedback, but processing the data afterwards was labour-intensive because the relatively unstructured presentation of qualitative and quantitative entry fields. A better solution would be to provide a more structured table for feedback collection with fields for ratings and comments which would make it easier to parse large amounts of data.

All testers were organised within a shared Discord group, which served as a central hub for instructions, logistical updates, and issue reporting. This structure proved highly effective. Testers were able to report bugs, share videos, and surface problems in real time, while fixes could be communicated efficiently to the group. Importantly, testers were instructed not to share evaluative feedback within the group itself, in order to avoid biasing one another’s assessments. The 7-day testing period proved sufficient for testers to organise their feedback over multiple sessions.

Ordering, Constructing, and Shipping Prototypes

The construction of the prototypes benefited from the use of readily available and open-source components, which kept unit costs relatively low and allowed for rapid iteration. Devices were deliberately designed with removable back panels, simplifying assembly and modification during the testing phase.

However, component sourcing introduced significant operational risk. Parts were often ordered from international merchants, and availability could change unexpectedly. A notable example was the printer component going out of stock, forcing a significant delay of over a week, as well as the selection of a substitute that appeared identical but ultimately introduced firmware compatibility issues. While these issues were resolved via over-the-air updates, they demonstrated how supply chain variability can introduce hidden complexity and risk into prototype testing. In future,

Shipping also proved more complex than anticipated. Most units were shipped internationally, while a smaller number were delivered in person. Despite using express couriers such as DHL, delays occurred due to customs handling, particularly where packages were marked as gifts to reduce duties. In some cases, testers were required to pay customs fees themselves or provide additional documentation to release shipments, leading to further delays.

The learning point here is to anticipate that customs duties will be levied even if the package is framed as a gift, in which case it is better to pre-pay this to avoid delays and cost to the recipient.

Outside of the delays from sourcing the out-of-stock printing component and customs handling, the other actions were performed within the expected timeframe.

Assisting Testers Through Challenges

The availability of over-the-air updates was critical to the success of the testing process. It allowed issues to be addressed quickly and enabled testers to continue evaluating intended features without interruption. Most reported issues were responded to within 24 hours, which helped maintain goodwill and momentum.

The shared Discord environment significantly reduced duplication of effort. Testers could see whether issues had already been identified and resolved, and guidance could be reused rather than repeated individually. Testers were generally patient and understanding, recognising that the system was a prototype and that issues were expected.

Supporting materials included a demonstration video and an EPUB help document outlining key processes. In hindsight, these resources should have been available from the outset, as earlier access would have reduced friction during initial setup and early testing stages.

Suggestions for Hardware Projects at Prototype Stage

Operational suggestions

This testing experience highlights the importance of aligning study scope with prototype maturity. Lightweight, expert-led testing is well suited to early-stage hardware, where the goal

is directional learning rather than validation. However, even lightweight studies benefit from careful attention to scale design, feedback standardisation, and operational risk.

Hardware projects should anticipate supply chain variability, regional logistics, and component substitutions as inherent risks. Designing prototypes to be modular, serviceable, and remotely updateable significantly reduces the cost of responding to these challenges.

It is also important to ensure that logistical and regional considerations are factored in at an early stage, such as providing the appropriate power plug/cable, and accounting for courier costs together with customs fees.

Finally, shared communication channels and clear separation between logistical discussion and evaluative feedback help maintain both efficiency and methodological integrity during testing.

Design suggestions

- **View prototypes as instruments for learning**

The value of a prototype lies as much in what it reveals as in what it delivers.

Instrumentation, observation, and structured feedback can help ensure that testing informs iteration rather than simply confirming assumptions.

- **Consider gating core functionality behind readiness checks**

In early prototypes, allowing users to access key workflows before prerequisites are met can amplify confusion. Introducing lightweight readiness checks may help prevent downstream errors and improve first-use confidence, particularly during setup and early interaction.

- **Aim to make system state visible wherever possible**

Ambiguity around whether a system is idle, processing, or stalled was a recurring source of friction. Providing clearer visual or audible indicators of state changes can reduce uncertainty, especially in asynchronous or network-dependent operations.

- **Treat perceived responsiveness as a design consideration, not just a performance metric**

Even when underlying processes take time, early acknowledgement of user input can materially improve trust. Prototypes may benefit from explicitly designing for perceived responsiveness alongside actual execution speed.

- **Be mindful of silent failure modes**

When errors occur without feedback, users often assume instability or malfunction. Making failure states more explicit, even when recovery options are limited, can help set expectations and reduce trial-and-error behaviour during testing.

- **Design interactions for realistic, imperfect conditions**

Prototype testing often surfaces issues that do not appear in ideal lab environments. Designing touch targets, inputs, and physical interactions with imperfect input and varied contexts in mind can improve robustness and reduce avoidable friction.

- **Where possible, align system language with users' real-world mental models**

Feedback suggested that technical terminology can increase cognitive load when it

diverges from how users think about tasks. Adapting labels and workflows to better reflect real-world usage may improve clarity without changing underlying functionality.

- **Favour recognition over recall in interface design**

When users are required to remember prior states, hidden context, or implicit rules, usability tends to suffer. Making context visible, such as showing full selections or histories, can lower cognitive effort, particularly in transactional flows.

- **Try to avoid presenting affordances that are not actionable**

Visual elements that appear interactive but cannot be used in a given state often lead to confusion. Aligning visual affordance with functional availability can help reduce misinterpretation and unnecessary retries.

- **Pay particular attention to the efficiency of repeated actions**

In prototype testing, inefficiencies tend to become more visible once users move beyond first-time exploration. Reducing unnecessary confirmations or steps in common workflows may improve perceived maturity and usability.

- **Differentiate between functional correctness and perceived correctness**

Several issues arose not from incorrect outputs, but from delays, presentation issues, or UI instability that undermined trust. Prototypes may benefit from considering how correctness is communicated, not just whether it is technically achieved.

- **Treat hardware ergonomics as part of the user experience**

Physical attributes such as size, edge treatment, weight distribution, and component placement were closely tied to usability. Hardware design decisions can meaningfully shape interaction patterns and should be considered alongside software behaviour.

- **Assume that recovery will be necessary and design accordingly**

Errors and crashes are often inevitable at prototype stage. Supporting safe, rapid recovery without data loss may be more valuable than attempting to prevent all failure cases upfront.

- **Be cautious of breaking core flows through external dependencies**

Where users must leave the device or system to confirm success, the experience can feel fragmented. Even provisional or partial in-system confirmation can help maintain a sense of flow during testing.

- **Prioritise consistency over minimalism**

Feedback suggested that inconsistency was more disruptive than visual density. Establishing consistent patterns for navigation, feedback, and interaction can reduce cognitive load, even in relatively simple interfaces.