Enabling On-Device Terminal Functionality on the reMarkable Paper Pro via its E-Ink Interface: A Technical Assessment

I. Introduction: The Quest for Terminal Access on reMarkable Paper Pro

A. The reMarkable Paper Pro: A Device for Focused Work Meets Technical Curiosity

The reMarkable Paper Pro has been introduced as a premium digital paper tablet, engineered primarily to provide a distraction-free environment for note-taking, reading, sketching, and focused work. It features an 11.8-inch Canvas Color display, which is based on E Ink Gallery™ 3 technology, and incorporates an adjustable reading light, aiming to deliver an experience that closely mimics traditional paper. The device runs reMarkable OS, a custom Linux distribution, a characteristic that inherently attracts technically proficient users. For this segment of users, the underlying Linux-based operating system presents an opportunity to explore and extend the tablet's capabilities beyond its stock functionalities. A common aspiration among such users is to gain direct command-line interface (CLI) access, or terminal usage, on the device itself. This desire stems from the potential to leverage the device for development, system administration, or simply to gain deeper control over its software environment. The pursuit of this functionality underscores a common theme in the realm of specialized electronic devices: the drive by a dedicated user base to unlock the full potential of their hardware.

B. Defining "On-Device Terminal Usage" via the E-Ink Interface

It is crucial to delineate the specific scope of "on-device terminal usage" as addressed in this report. The focus is on the ability to run a terminal emulator application *directly on the reMarkable Paper Pro's e-ink screen*, thereby enabling command-line interaction natively on the tablet. This is fundamentally different from accessing the Paper Pro's shell via Secure Shell (SSH) from an external computer (e.g., a laptop or desktop). While SSH access provides command-line control over the tablet's operating system, the interaction occurs on the external machine's display. The user query guiding this research specifically seeks information about terminal usage "via the eink interface," indicating a clear interest in this native, on-screen command-line experience. This distinction is paramount, as official support exists for SSH access, but native e-ink terminal functionality falls into the realm of community-driven modifications.

C. Scope and Methodology of this Report

This report synthesizes available information from official reMarkable documentation, community forums (such as Reddit), developer repositories (primarily GitHub), and documented user experiences to provide a comprehensive assessment of the feasibility and current status of achieving on-device terminal functionality on the reMarkable Paper Pro. The reMarkable Paper Pro is often identified by its internal codename "ferrari" , and this term will be used interchangeably where relevant. The analysis covers the period up to early 2025, reflecting the most current information found within the provided research materials. The investigation delves into both officially provided developer resources and the unofficial, community-led efforts to extend the tablet's capabilities. The nature of the user's request for "Deep Research" necessitates a thorough exploration, moving beyond superficial findings to investigate the technical nuances, practical challenges, community dynamics, and potential future trajectories of this endeavor. This involves not only identifying *if* on-device terminal usage is possible but also *how* it might be achieved, the tools involved, the inherent difficulties, and the broader implications for users and the device ecosystem.

II. Official Channels: reMarkable's Stance and Developer Provisions

While reMarkable's primary marketing and product design emphasize a curated, distraction-free user experience, the company has made specific provisions for developers and technically advanced users, particularly with the Paper Pro model. These provisions are foundational for any attempt to run custom software, including an on-device terminal.

A. Developer Mode on reMarkable Paper Pro

A significant official feature for the reMarkable Paper Pro is "Developer Mode". This mode is exclusive to the Paper Pro and is not officially available for previous reMarkable models like the reMarkable 1 or reMarkable 2. Enabling Developer Mode grants users root shell access to the device's operating system via the SSH protocol.
The process of enabling Developer Mode is not without consequence. It requires a factory reset of the tablet, which means all user data, notes, and documents stored on the device will be erased. Users are explicitly warned about this data loss and must accept the associated risks before proceeding. Furthermore, reMarkable cautions that any errors, defects, or malfunctions arising from modifications made while in Developer Mode may not be covered by the standard warranty or any protection plans. Once enabled, the device will display a warning message on every boot, informing the user that Developer Mode is active; this warning cannot be removed or modified as it is part of the early boot process still protected by secure boot.
After Developer Mode is activated, SSH access credentials – typically the username root and a randomly generated password – can be found within the device's settings menu, under General > Help > About > Copyrights and licenses. Connectivity for SSH is primarily established via a USB connection, which creates an ethernet-over-usb network interface. The tablet is typically accessible at the IP address 10.11.99.1 over this connection. Additionally, SSH access over Wi-Fi can be enabled by executing the command rm-ssh-over-wlan on in an SSH session connected via USB.

B. Official Software Development Kit (SDK) and Source Code

We should set rm-ssh-over-wlan on at boot.

Complementing Developer Mode, reMarkable provides an official Software Development Kit (SDK) specifically for the Paper Pro (identified by the codename "ferrari"). This SDK is essential for developers who wish to cross-compile software to run natively on the Paper Pro. The SDK includes a cross-compiler toolchain, shared libraries, and header files necessary for building applications that target the Paper Pro's ARM64 architecture (specifically its NXP iMX8 Mini processor).
In addition to the SDK, reMarkable also makes available the Linux kernel source code for the Paper Pro. This source, found in repositories like reMarkable/linux-imx-rm, is based on the NXP iMX platform and is tagged for specific reMarkable OS versions. The U-boot (bootloader) source code is also provided. Access to the kernel and bootloader source code allows for a deep understanding of the system and enables advanced customizations, such as building custom kernel modules or modifying boot parameters, though these are generally beyond the scope of simply running a user-space terminal application.

C. Absence of an Official On-Device E-Ink Terminal

Despite these developer-centric provisions, official reMarkable documentation, product feature lists, and marketing materials for the Paper Pro do not mention any native application or built-in feature for using a terminal directly on the device's e-ink display. The explicit focus of Developer Mode and the associated tools is to facilitate SSH access *to* the device for development and debugging purposes, not to provide an on-screen terminal *on* the device.
The official stance and provisions highlight a clear distinction: reMarkable enables deep system access and provides the tools for custom software development (Developer Mode, SDK, kernel source) but does not directly support or offer an on-device e-ink terminal as a feature. This implies that while the *foundations* for such functionality are laid by the manufacturer, the actual implementation of an e-ink terminal must come from unofficial, community-driven efforts. This dynamic positions reMarkable as a facilitator of low-level access for its most technical users, while leaving the development of certain advanced, user-facing tools to the ingenuity and persistence of the third-party developer community. Consequently, users seeking an on-device terminal experience must turn to these community solutions, which inherently come with their own set of development challenges, potential risks, and ongoing maintenance considerations.

Table 1: Official reMarkable Paper Pro Developer Resources

Resource	Key Features	Relevance to Customization/Terminal Development	Official Source/Link (Examples)
Developer Mode	SSH access, root privileges, factory reset required, Paper Pro only, boot warning.	Enables modification of the OS and installation of custom software, including terminal emulators or their dependencies.	
SDK ("ferrari")	Cross-compiler toolchain (ARM64), libraries, headers for Paper Pro (NXP iMX8	Essential for compiling native applications like C-based terminal emulators or	

Resource	Key Features	Relevance to Customization/Terminal Development	Official Source/Link (Examples)
	Mini).	framebuffer utilities specifically for the Paper Pro.	
Linux Kernel Source	linux-imx-rm for "ferrari".	Allows understanding and potential modification of the kernel, relevant for custom drivers or advanced framebuffer interaction.	
U-boot Source	Bootloader source code.	For advanced customization of the boot process; less directly relevant for a user-space terminal but part of the open development ecosystem.	

This table summarizes the official resources that form the starting point for any developer looking to customize the reMarkable Paper Pro or develop applications like an on-device terminal. These tools provide the necessary access and build environments upon which community efforts are constructed.

III. Community Endeavors: Hacking a Terminal onto the E-Ink Screen

The absence of an official on-device terminal for the reMarkable Paper Pro has not deterred the technically inclined user base. A vibrant and resourceful community has emerged, dedicated to exploring, modifying, and extending the capabilities of reMarkable tablets. These efforts, while unofficial, are crucial for realizing functionalities like an e-ink terminal.

A. The Vibrant reMarkable Modding Community

The reMarkable modding community is characterized by a collaborative spirit, sharing knowledge, tools, and custom projects through various online platforms. Several key hubs serve as focal points for these activities:

- **remarkable.guide**: This website is a comprehensive, community-maintained resource offering guides and information on accessing device internals, installing custom software, and general modding for reMarkable tablets.
- **Discord Servers**: Platforms like Discord host active communities where users and developers engage in real-time discussions, provide support, and share progress on various projects. While direct searches for "Paper Pro terminal" within Discord archives did not yield specific, actionable results in the provided materials, these servers remain

valuable for general reMarkable hacking discussions.

- **awesome-reMarkable (GitHub)**: This curated GitHub repository lists a wide array of tools, applications, and resources developed by the community for reMarkable tablets. While direct mentions of terminal emulators specifically packaged and confirmed for the Paper Pro are sparse in the snippets, it serves as an excellent discovery tool for related projects.

B. Learnings from reMarkable 2: Paving the Way (and Highlighting Challenges)

Much of the foundational work for running custom applications, including terminals, on reMarkable devices was initially focused on the reMarkable 2 (rM2). These efforts provide valuable insights and a potential blueprint for similar endeavors on the Paper Pro, while also highlighting common challenges.

1. Toltec Package Manager: Toltec emerged as a critical piece of infrastructure for the rM2 modding scene. It is a community-maintained software repository providing an opkg-based package management system, simplifying the installation of third-party software, including terminal emulators and their dependencies. However, a significant challenge with Toltec has been maintaining compatibility with the latest official reMarkable OS updates. Historically, Toltec's support often lagged, with specific versions being tied to older OS releases (e.g., OS v2.15). Given that newer devices like the Paper Pro ship with more recent OS versions (v3.x and later) , this OS version dependency presents a major hurdle for leveraging Toltec directly on the Paper Pro without updated support. Discussions regarding Toltec's adaptation for newer OS versions and specific device models, including the Paper Pro, are ongoing within the community.

2. Framebuffer Terminals (e.g., yaft) Lightweight framebuffer terminals, such as yaft (Yet Another Framebuffer Terminal), have been successfully run on the rM2. These applications bypass the standard graphical user interface (Xochitl) and render directly to the e-ink display's framebuffer. The rM2-stuff repository on GitHub, maintained by user "timower," has been a key source for yaft and related tools for the rM2.

3. Framebuffer Utilities (e.g., rm2fb) For framebuffer terminals like yaft to function correctly on the rM2, a utility to manage the e-ink display's specifics is often required. rm2fb is one such custom framebuffer implementation for the rM2, offering lower-level hooking capabilities and compatibility with newer versions of Xochitl (the main reMarkable UI application). Such utilities are essential for abstracting the hardware-specific details of the e-ink display controller.

4. Keyboard Input (e.g., keyd, Type Folio) Effective terminal usage necessitates a physical keyboard. The reMarkable Type Folio, available for both rM2 and Paper Pro , is the primary candidate. However, integrating its input into custom applications like a terminal requires system-level handling. keyd, a keyboard mapping daemon, has been used on the rM2 to correctly interpret key events from the Type Folio and other external keyboards.

C. Targeting the reMarkable Paper Pro ("ferrari")

Adapting these rM2 solutions or developing new ones for the Paper Pro involves addressing its unique hardware and software environment.

1. Current Status of Toltec for Paper Pro: The compatibility of Toltec with the Paper Pro (codename "ferrari") is a critical factor for ease of software installation. As of late 2024, discussions within the Toltec community indicated active interest and investigation into Paper

Pro support. However, the persistent challenges of Toltec keeping pace with reMarkable's OS updates and underlying system library changes (like glibc) likely extend to the Paper Pro. This means that stable, comprehensive Toltec support for the latest Paper Pro OS versions may not be immediately available.

2. RMPP Entware and Alternative Package Management: An alternative approach for package management on the Paper Pro has emerged with RMPP Entware. This project, mentioned in the awesome-reMarkable list, aims to install the opkg package manager specifically for the reMarkable Paper Pro. If Toltec's full support for the Paper Pro is delayed due to the complexities of OS updates and new hardware, RMPP Entware could offer a more direct, albeit potentially more manual, pathway for installing essential Unix utilities and, by extension, terminal emulators. This approach might bypass some of the broader compatibility testing and integration work required for full Toltec support, potentially providing a quicker route for technically adept users to install individual packages.

3. Framebuffer Solutions for E Ink Gallery 3: The Paper Pro's E Ink Gallery™ 3 color display is a significant departure from the monochrome displays of previous models. This new display technology has different characteristics regarding color planes, refresh mechanisms, and controller interactions. Consequently, framebuffer utilities like rm2fb, which were designed for the rM2's monochrome display , are unlikely to function correctly on the Paper Pro without substantial adaptation or a complete rewrite. There is no clear evidence in the provided materials of a direct port or a readily available community-vetted, C-based framebuffer utility specifically for the Paper Pro's Gallery 3 display that would enable applications like yaft to run out-of-the-box. This represents a key technical hurdle.

4. Reported Successes and Ongoing Projects: While specific, easily replicable guides for an on-device e-ink terminal on the Paper Pro are not prominent in the research, there are indications of interest and experimentation:

- User Sam Littlefair reported successfully hacking a reMarkable tablet (model not explicitly stated as Paper Pro in the snippet, but the context implies general reMarkable hacking) to run a terminal emulator and the Helix editor. However, detailed software or installation guides were not provided in that specific post.
- A Reddit post titled "Terminal on the Remarkable Paper Pro" initially seemed promising. However, the associated discussion and video link clearly demonstrate yaft running on a reMarkable 2 with a Type Folio, not a Paper Pro. The title may have been aspirational or a misunderstanding at the time of posting.
- The reMarkaDOS project enables users to run FreeDOS (which includes a terminal environment) on reMarkable tablets via the BOCHS emulator. While this demonstrates running a terminal-like interface, it's within an emulated DOS environment, not a native Linux terminal interacting directly with the reMarkable OS and its e-ink display. This project also relies on Toltec for installation.
- A GitHub issue related to goMarkableStream, an application originally designed for rM2, discussed the challenges of porting it to the Paper Pro (RMPP), noting issues like hardcoded screen resolutions and the need to recompile for the Paper Pro's specific System-on-Chip (SoC). This exemplifies the general porting difficulties faced when moving from rM2 to Paper Pro.

D. Alternative Development Paths: python-libremarkable

A potentially promising avenue for developing custom e-ink applications, including terminals, on the Paper Pro is the python-libremarkable library. This Python library offers extensive

functionalities for interacting directly with the reMarkable framebuffer. Its capabilities include drawing text, setting colors, defining screen regions, and managing e-ink refresh waveform modes. Crucially, the library explicitly mentions support for color values and names in its API , making it theoretically well-suited for the Paper Pro's E Ink Gallery 3 display.
The availability of python-libremarkable could lower the barrier to entry for creating custom e-ink applications on the Paper Pro, especially for developers proficient in Python. While developing a robust, C-based framebuffer utility for a new display technology like Gallery 3 can be a complex and time-consuming undertaking, python-libremarkable provides higher-level abstractions that might allow for faster prototyping and development of a functional e-ink terminal. Its functions for text drawing (draw_text, draw_multiline_text) and precise control over refresh modes are directly applicable to building a terminal interface. However, potential performance differences between a Python-based solution and a native C-based one would need to be considered for an application as interactive as a terminal.

Table 2: Key Community Software/Tools for reMarkable Terminal Emulation

Tool	Description	Current Status/Compatibility for Paper Pro ("ferrari")	Key Community Links/Sources (Examples)
Toltec	Community package manager (opkg) for reMarkable.	Support for Paper Pro ("ferrari") actively discussed ; historically challenged by OS updates. Full, stable support for latest Paper Pro OS unconfirmed.	toltec-dev.org
RMPP Entware	Entware (opkg) installer specifically for Paper Pro.	Mentioned as available. May offer a more direct route for installing packages on Paper Pro if Toltec support is pending.	GitHub: hmenzagh/rmpp-entware (from)
yaft	Lightweight framebuffer terminal emulator.	Works on rM2. No direct confirmation of a stable, easily installable version for Paper Pro e-ink. Requires a Paper Pro-specific framebuffer solution.	GitHub: timower/rM2-stuff
rm2fb	Framebuffer utility for rM2.	Designed for rM2's monochrome display; unlikely to work on Paper Pro's E Ink Gallery 3 without significant adaptation or replacement.	GitHub: timower/rM2-stuff

Tool	Description	Current Status/Compatibility for Paper Pro ("ferrari")	Key Community Links/Sources (Examples)
keyd	Keyboard mapping daemon.	Likely adaptable for Paper Pro if input event system is similar to rM2; used with Type Folio on rM2. keyd_2.4.3-1_rmall.ipk installable via opkg.	GitHub: rvaiya/keyd (from)
python-libremarkable	Python library for direct framebuffer access.	Appears highly suitable for Paper Pro due to documented color support and comprehensive framebuffer control functions. Actively developed.	libremarkable.eeems.codes

This table outlines the key software components that are part of the community's toolkit for enabling advanced functionalities like terminal emulation. It underscores that achieving a terminal on the Paper Pro is not a matter of installing a single application but rather relies on an ecosystem of tools, some of which require specific adaptation or development for the new hardware and software environment of the "ferrari" model.

IV. Technical Deep Dive: The Path to an E-Ink Terminal on "ferrari"

Achieving functional on-device terminal usage on the reMarkable Paper Pro ("ferrari") requires a technical understanding of its specific hardware, operating system, and particularly its novel E Ink Gallery 3 display.

A. Paper Pro ("ferrari") Hardware and OS Specifics

The reMarkable Paper Pro presents a distinct hardware and software profile compared to its predecessors:

- **Display:** The most notable feature is its 11.8-inch Canvas Color display, based on E Ink Gallery™ 3 technology. It offers a resolution of 2160 x 1620 pixels (229 PPI). This color e-ink panel is a fundamental differentiator from the monochrome displays of the reMarkable 1 and 2, introducing new complexities for direct display manipulation. E Ink Gallery 3 technology itself has specified update times that vary for black and white versus different color modes, and it supports pen input.
- **Processor:** The Paper Pro is powered by an NXP iMX8 Mini System-on-Chip (SoC), which features a 1.8 GHz quad-core Cortex-A53 processor. This is an ARM64 architecture. This differs from the processors in earlier reMarkable models, necessitating recompilation of any native C/C++ code. The official reMarkable SDK for "ferrari" targets this ARM64 architecture.

- **Operating System:** The device runs reMarkable OS (internally codenamed Codex), which is a custom Linux distribution built using the Yocto Project. The availability of the Linux kernel source (linux-imx-rm) provides insight into the OS's low-level workings.
- **Storage and RAM:** The Paper Pro is equipped with 64 GB of internal storage and 2 GB of LPDDR4 RAM , offering more resources than previous models.

B. Interfacing with the Framebuffer on E Ink Gallery 3

For custom applications to render directly on the e-ink screen, interaction with the Linux framebuffer (typically /dev/fb0) is the standard approach. However, E Ink Gallery 3 introduces specific considerations:

- **Refresh Modes (Waveforms):** E Ink displays, especially color ones, rely on complex voltage sequences, known as waveforms, to update the screen. These waveforms control how pixels transition, affecting update speed, color accuracy, and the presence of artifacts like ghosting. E Ink Gallery 3 has distinct update characteristics for its various color modes and black-and-white mode. Effective management of these waveforms is critical for a usable terminal experience. The python-libremarkable library provides an API to select different waveform modes , which is a crucial capability.
- **Driver Interaction:** While Linux provides a generic framebuffer interface (/dev/fbX), the device-specific driver for the E Ink Gallery 3 controller handles the translation of framebuffer data into the actual e-ink update sequences. The Uno Platform's documentation on Linux Framebuffer support notes that the driver provides physical information that can be used for DPI scaling, but the core interaction is with the framebuffer device itself.
- **Complexity of Color E-Ink:** The E Ink Gallery 3 display, while offering color, introduces a higher level of complexity in management compared to monochrome e-ink. It has multiple color modes, each with different update speeds: black and white updates are the fastest (around 350ms), while various color modes can take from 500ms to 1500ms for optimal quality. A terminal application needs relatively rapid screen updates for user interactivity. Achieving this on Gallery 3, especially if color is used (e.g., for syntax highlighting), requires careful selection of refresh waveforms and strategic use of partial updates to minimize perceived latency and avoid excessive ghosting. Standard Linux framebuffer access provides the raw data path, but the e-ink controller's driver ultimately dictates how these updates are rendered. Libraries like python-libremarkable aim to abstract some of this complexity, but achieving optimal performance for a terminal might necessitate very specific sequences of operations or even direct ioctl calls to the framebuffer driver if higher-level abstractions prove insufficient. This means developers working on an e-ink terminal for the Paper Pro must either deeply understand these E Ink Gallery 3 specifics or rely on a well-optimized library that correctly handles its unique refresh characteristics. Simple, full-screen refreshes common in basic framebuffer applications will likely be too slow or visually disruptive for a satisfactory terminal experience.

C. Software Compilation, Deployment, and OS Integration

Any native terminal emulator (typically written in C or C++) must be cross-compiled for the Paper Pro's ARM64 architecture using the official SDK. Once compiled, the binaries need to be transferred to the device, commonly via SCP after enabling SSH access through Developer Mode. Correct execution permissions must then be set.

Integrating custom applications with the main reMarkable user interface, Xochitl , can be challenging. Third-party launchers like Oxide or Draft (listed in awesome-reMarkable) are often employed on older models to manage and run custom applications. Alternatively, a terminal application could be initiated from an SSH session and then take control of the framebuffer for its display.
Modifying the root filesystem, for instance, to install software into standard paths like /usr/bin or /opt, requires remounting the root partition as read-write (e.g., using the command mount -o remount,rw /). This change is typically not persistent across reboots. For persistent changes to system configuration files in directories like /etc, it might be necessary to unmount an overlay filesystem that protects these areas.

D. Input Methods: Type Folio and On-Screen Keyboards

For practical terminal usage, a physical keyboard is indispensable. The reMarkable Paper Pro has an official Type Folio accessory that connects via a magnetic port and includes backlit keys. Ensuring that key events from the Type Folio are correctly interpreted by a custom terminal application would likely involve using a keyboard daemon like keyd to manage input mappings, or by directly reading and processing events from the relevant Linux input device (e.g., /dev/input/eventX).
While an on-screen keyboard could theoretically be implemented to render via the framebuffer, it would be cumbersome for extensive terminal work. The native reMarkable OS does provide an on-screen keyboard for quick text input within its applications. The terminal emulator yaft has been noted to work with an on-screen touch-based keyboard on the reMarkable 2 , suggesting that such an interface could be developed for the Paper Pro if desired, though its utility for serious command-line work would be limited.

Table 3: reMarkable Paper Pro vs. reMarkable 2: Key Differences for Terminal Development

Feature	reMarkable 2	reMarkable Paper Pro ("ferrari")
Display Technology	Monochrome E-Ink	Canvas Color (E Ink Gallery™ 3)
Processor Architecture	ARM32 (e.g., i.MX6 SoloLite or similar)	ARM64 (NXP iMX8 Mini Quad-core Cortex-A53)
Official Developer Access	Root access primarily via community methods/hacks	Official Developer Mode (SSH, root access)
Known Framebuffer Utility	rm2fb (community-developed)	No specific C-based utility confirmed for Paper Pro; python-libremarkable is a strong candidate.
Primary SDK Target	rm2	ferrari
Main UI Application	Xochitl	Xochitl

This table highlights the crucial distinctions between the reMarkable 2 and the Paper Pro. These differences, particularly in display technology and processor architecture, mean that solutions developed for the rM2 cannot be directly applied to the Paper Pro. New or significantly adapted development efforts are necessary to achieve similar functionalities, like an on-device terminal, on the "ferrari" model.

V. Challenges, Risks, and E-Ink Performance Considerations

Embarking on the path to enable on-device terminal usage on the reMarkable Paper Pro is not without its hurdles. Users and developers must navigate software compatibility issues, potential device risks, and the inherent performance characteristics of e-ink displays.

A. OS Versioning and Software Compatibility ("The Cat and Mouse Game")

reMarkable, like many device manufacturers, periodically releases updates to its operating system, reMarkable OS. While these updates often bring new features and improvements, they can inadvertently break community-developed tools, hacks, and custom modifications. This creates an ongoing "cat and mouse game" where the community must adapt its software to each new OS version.
Maintaining compatibility for foundational tools like the Toltec package manager or custom framebuffer utilities across different reMarkable OS versions is a significant and continuous effort for community developers. The process of downgrading the OS to an older, compatible version—a strategy sometimes employed on the rM2 to enable certain hacks —is inherently risky. Downgrading can lead to system instability, may not always be feasible on newer hardware revisions, and could result in the loss of official features or compatibility with accessories like the Type Folio, which often requires newer firmware versions (e.g., Type Folio needing v3.x, while older Toltec builds were for v2.x).

B. Potential Risks: Bricking, Warranty, and Data Loss

Modifying the system software of any embedded device, including the reMarkable Paper Pro, carries inherent risks. Enabling Developer Mode and installing third-party software or custom modifications can, in worst-case scenarios, lead to a "soft-bricked" device—a state where the device fails to boot or operate correctly. Recovering from such a state can be complex and may require advanced technical skills.
reMarkable's official stance is that any errors, defects, or malfunctions caused by using Developer Mode or making unauthorized software modifications are not covered by the standard product warranty or any extended protection plans. Users proceed with such modifications at their own risk. Furthermore, the process of enabling Developer Mode on the Paper Pro mandates a factory reset, which results in the complete erasure of all user data stored on the device. It is imperative for users to back up their data before initiating this process.

C. E-Ink Display Dynamics for Terminal Use

The E Ink display, while offering excellent readability and low power consumption, presents unique challenges for dynamic applications like a terminal:

- **Responsiveness:** E-ink panels have inherently slower refresh rates compared to traditional LCD or OLED screens. While the reMarkable Paper Pro boasts a relatively fast

e-ink latency of 12ms , achieving a "snappy" and interactive terminal feel still requires highly optimized refresh strategies. This includes the careful use of partial screen updates and the selection of appropriate e-ink waveforms to minimize perceived lag during typing and scrolling.
- **Ghosting:** Improper or infrequent screen refreshes on e-ink displays can lead to "ghosting," where faint remnants of previous screen content remain visible. This can be particularly distracting and detrimental to readability in a terminal environment where text changes rapidly.
- **Color on Gallery 3:** The Paper Pro's E Ink Gallery 3 display introduces color capabilities. While this opens possibilities for features like syntax highlighting in a terminal, it also adds complexity. E Ink Gallery 3 has different update times for its various color modes compared to its faster black-and-white update mode. For instance, black and white updates are around 350ms, whereas color updates can range from 500ms to 1500ms depending on the desired quality. Developers must decide whether to prioritize the speed of monochrome updates for general terminal interactivity or to leverage color at the cost of potentially slower refresh rates for specific elements.
- **Readability and Contrast:** Although e-ink is generally well-suited for text, font rendering, contrast levels, and anti-aliasing techniques need to be optimized for a terminal interface to ensure clarity and reduce eye strain during prolonged use.
- **Animations and Effects:** Modern desktop environments often incorporate animations and graphical effects that perform poorly on e-ink displays, leading to ghosting and visual artifacts. While a text-based terminal is less prone to these issues, it still requires clean and efficient screen refreshes.

The unique characteristics of e-ink necessitate a careful balance. The paper-like, eye-friendly nature of the display is a significant advantage, potentially offering a comfortable environment for focused coding or system administration tasks within a terminal. However, the demands of a highly interactive terminal—requiring quick feedback for typed characters, smooth scrolling, and minimal visual artifacts—clash with the inherent refresh limitations of current e-ink technology. A successful e-ink terminal on the Paper Pro must therefore be meticulously optimized. It will likely not match the raw performance of a terminal on a conventional LCD or OLED screen, but with careful software engineering, it could become a highly usable tool for specific tasks, particularly those benefiting from the focused, low-glare environment that reMarkable devices provide. Users approaching this functionality should have realistic expectations regarding its performance characteristics.

VI. Current Status (Early 2025) and Future Outlook

Assessing the landscape of on-device e-ink terminal usage for the reMarkable Paper Pro as of early 2025 reveals a mix of established official capabilities and evolving community-driven initiatives. The user's query about work "being done or currently working" reflects a desire for up-to-date information in this dynamic space.

A. Confirmed Methods vs. Experimental/In-Progress for Paper Pro

- **Confirmed and Official:** Accessing a root shell on the reMarkable Paper Pro via SSH after enabling Developer Mode is a well-established and officially documented procedure. This provides full command-line access to the device's underlying Linux operating system from an external computer.

- **Experimental, In-Progress, or Unclear for On-Device E-Ink Terminal:**
  - **Direct E-Ink Terminal (e.g., yaft):** While yaft and similar framebuffer terminals have been successfully demonstrated on the reMarkable 2 , a readily available, easily installable, and community-vetted solution specifically for the Paper Pro's E Ink Gallery 3 display is not clearly evident from the analyzed materials. The primary technical barrier appears to be the need for a Paper Pro-specific framebuffer utility or library that can effectively manage the color e-ink display.
  - **Toltec Package Manager for "ferrari":** Support for the Paper Pro (codename "ferrari") within the Toltec package manager is a subject of ongoing discussion and development within the community. However, Toltec's ability to keep pace with reMarkable's frequent OS updates remains a persistent challenge. The readiness of Toltec to provide a stable and comprehensive repository of packages, including terminal emulators, for the latest Paper Pro OS versions is a key dependency.
  - **RMPP Entware:** The RMPP Entware project suggests a more direct method for installing the opkg package manager on the Paper Pro. This could potentially enable users to manually install terminal emulators and their dependencies, provided those dependencies can be met on the Paper Pro's OS and hardware.
  - **python-libremarkable:** This Python library offers the necessary building blocks (framebuffer access, text drawing, color manipulation, refresh control) for developing a Python-based e-ink terminal on the Paper Pro. However, a complete, packaged, and widely adopted terminal application built upon this library is not explicitly mentioned as being available.
  - **Individual Efforts:** Anecdotal reports of individuals successfully running terminal emulators on reMarkable devices exist (e.g., Sam Littlefair), but detailed, publicly available guides or reproducible projects specifically for the Paper Pro's e-ink interface are not prominent in the provided information.

B. Key Communities and Repositories to Monitor

For users interested in tracking progress or contributing to efforts to enable on-device terminal usage on the Paper Pro, the following community hubs and repositories are crucial:

- **Toltec Development:** The toltec-dev GitHub organization (particularly its issue trackers and discussion forums) is the primary place to monitor progress on Toltec support for the "ferrari" model.
- **awesome-reMarkable:** The reHackable/awesome-reMarkable GitHub repository serves as a curated list and discovery tool for new community projects and tools related to reMarkable tablets.
- **remarkable.guide and Associated Discord:** This community-maintained website and its linked Discord server are valuable resources for guides, discussions, and real-time support from other reMarkable modders.
- **Specific Developer Repositories:** Progress on Paper Pro-specific framebuffer tools or terminal applications will likely first appear in individual developer repositories on platforms like GitHub. Monitoring projects that build upon the Paper Pro SDK or target its unique hardware (e.g., if projects similar to timower/rM2-stuff emerge for "ferrari") would be beneficial.
- **python-libremarkable Development:** The Eeems-Org/python-libremarkable project and its associated documentation/community channels are key for those interested in

Python-based e-ink application development for the Paper Pro.

C. Outlook for 2024-2025

The user's query emphasizes work being done or currently working, implying a timeframe relevant to recent and ongoing developments. The information gathered largely reflects activities and discussions up to early 2025. It is important to note that the codename "ferrari" for the Paper Pro is an internal reMarkable designation and is unrelated to any automotive manufacturer's product plans for 2025.
The path to a polished, easy-to-install e-ink terminal on the reMarkable Paper Pro is an evolutionary one, heavily dependent on the dedication and expertise of community developers. While the foundational elements are largely in place—Developer Mode providing root access, an official SDK for compilation, the underlying Linux OS, and emerging package management efforts like RMPP Entware—the critical missing piece for leveraging popular C-based terminals like yaft appears to be a robust, Paper Pro-specific framebuffer handling solution for the E Ink Gallery 3 display. Python-based approaches using python-libremarkable might yield functional results more quickly due to the library's higher-level abstractions for display control.
Progress in this area is likely to be incremental, with solutions potentially appearing first in niche GitHub repositories or being discussed in specialized Discord channels before achieving broader adoption or packaging through systems like Toltec (if and when its Paper Pro support matures). The timeline for such developments is inherently unpredictable as it relies on volunteer contributions and the community's ability to overcome technical hurdles. Therefore, users anticipating an immediate, off-the-shelf e-ink terminal solution for the Paper Pro in early 2025 might find the current options limited or highly experimental. However, the active community, the availability of developer tools from reMarkable, and the clear interest in this functionality suggest that such solutions are feasible and likely to emerge or mature over time. Patience and active monitoring of the aforementioned community channels will be key for those eager to utilize a terminal directly on their Paper Pro's e-ink screen.

VII. Conclusion and Recommendations

The investigation into enabling terminal usage directly on the reMarkable Paper Pro's e-ink interface reveals a landscape of official enablement for developers coupled with active, albeit challenging, community-driven innovation.

A. Summary of Findings

Officially, reMarkable provides robust tools for developers wishing to interact with the Paper Pro at a low level. Developer Mode grants essential root SSH access, and the official SDK allows for the compilation of custom software for the device's ARM64 architecture. However, an on-device terminal application that runs directly on the e-ink screen is not an officially supported feature. The realization of such functionality, therefore, falls entirely to the efforts of the reMarkable modding community. While significant precedents and tools exist from work done on the reMarkable 2 (such as the Toltec package manager and framebuffer terminals like yaft), the Paper Pro's distinct hardware—most notably its E Ink Gallery 3 color display and NXP iMX8 Mini processor—necessitates adapted or entirely new solutions. A critical component for running traditional framebuffer terminals is a device-specific utility or library to manage the

Since there is an available compilation toolchain for remarkable pro, I'd like to explore that route.

My requirements for refresh rate are reduced as I intend to interact with 'claude code' via the tablet.

I will primarily interact using on device handwriting conversion that I can paste into an input and submit to Claude code. 'Claude' CLI. Therefore as long as I eventually get a rendering of the output I can proceed to the next step.

Preferably I'd be able to interact with the output as ink. However, even if the output can be written to Google docs I can import to the tablet via the Google Drive integration. This would be a temporary workaround.

Since there is an available compilation toolchain for remarkable pro, I'd like to explore that route.

My requirements for refresh rate are reduced as I intend to interact with 'claude code' via the tablet.

I will primarily interact using on device handwriting conversion that I can paste into an input and submit to Claude code. 'claude' CLI. Therefore as long as I eventually get a rendering of the output I can proceed to the next step.

Preferably I'd be able to interact with the output as ink. However, even if the output can be written to Google docs I can import to the tablet via the Google Drive integration. This would be a temporary workaround as I prefer the flexibility of working with ink.

I'd like to do some research and development around creating a handwriting and/or text interface to Claude code.

This could take various forms. For example:

using rmapi to poll for cloud changes and processing with Claude's vision.

On device pasting of text from on device handwriting conversion.

I'll allow you to explore possibilities that will achieve my workflow goals.

1.) I would prefer to always work with ink so that web content and research can be cropped, moved, resized, duplicated, etc.

2.) Should be seamless
- Selecting a tag on a page would trigger the conversion to png for Claude to read and act on.

3.) Should facilitate real collaborite work with an AI persona that has all the abilities provided by Claude and available MCP server tools and thus providing them to the user via an E-ink interface. (There could be potential for a custom assistant control panel UI).

4) Ability to review, approve, and retain log of Claude's actions & output as editable ink.

5) Full knowledgebase archival and querying with neo4j. (all canvas & MCP output)

6) Index notebook
- kind of a glossery of NLP based and symantic links in knowledgebase. Provides summaries and cross references.

7) Beautiful
- include graphics, graphs, tables, and charts. All in editable ink that takes advantage of the remarkable Pro's display.

unique characteristics of the Gallery 3 display; such a C-based tool for the Paper Pro is not yet clearly established within the community.
Promising avenues include the RMPP Entware project for package management and the python-libremarkable library, which offers extensive control over the framebuffer, including color manipulation, making it a strong candidate for developing Python-based e-ink applications like a terminal. As of early 2025, a universally adopted, easy-to-install, and fully stable e-ink terminal solution specifically for the reMarkable Paper Pro is not yet apparent from the available data, but the foundational elements and strong community interest suggest ongoing development.

B. Guidance for Technically-Inclined Users

For users with the technical expertise and willingness to experiment, the following guidance is offered:

- **Prerequisites:** A solid understanding of Linux environments, command-line operations, SSH, and potentially C/C++ or Python programming and cross-compilation concepts is highly recommended.
- **Starting Point:** The first step is to enable Developer Mode on the reMarkable Paper Pro. Users must fully understand and accept the associated risks, including potential data loss and warranty implications.
- **Potential Pathways to Explore (with associated caveats):**
    1. **Monitor Toltec Development:** Keep a close watch on the toltec-dev community for official support for the Paper Pro ("ferrari"). If and when Toltec provides stable packages for the Paper Pro, it would likely be the most straightforward method for installing terminal emulators and other custom software.
    2. **Investigate RMPP Entware:** Explore installing opkg via the RMPP Entware project. This could allow for the manual installation of yaft or other terminal emulators, though careful attention to dependencies and compatibility with the Paper Pro's OS will be required. This path is more advanced and may involve significant troubleshooting.
    3. **Develop with python-libremarkable:** For users proficient in Python, experimenting with the python-libremarkable library offers a direct way to create custom e-ink applications, including a terminal interface. This provides granular control over the Gallery 3 display's features.
    4. **Adapt Existing C-based Solutions:** Highly advanced users could attempt to adapt existing C-based terminal emulators like yaft and framebuffer utilities like rm2fb (or develop a new Paper Pro-specific framebuffer library). This would involve using the official reMarkable Paper Pro SDK and potentially delving into the kernel source. This represents a substantial development effort.
- **Essential Tools:** A reliable SSH client on a host computer, the official reMarkable Paper Pro SDK (if compiling from source), a methodical approach to troubleshooting, and considerable patience are indispensable.

C. The Spirit of Innovation and Future Potential

The reMarkable community consistently demonstrates a strong drive to push the boundaries of their devices, transforming them into platforms that extend beyond their original design intent. The pursuit of an on-device terminal for the Paper Pro is a testament to this spirit. While significant technical challenges remain, particularly concerning the E Ink Gallery 3 display and

OS compatibility, the combination of official developer enablement from reMarkable for the Paper Pro and the ingenuity of the community makes the eventual emergence of more robust and accessible on-device terminal solutions probable.
The reMarkable Paper Pro, with its enhanced processing power, increased memory, and novel color e-ink display, presents an exciting canvas for such custom tools. A well-implemented e-ink terminal could offer a unique, focused environment for coding, system management, or text-based productivity, aligning with the device's core philosophy of distraction-free work while catering to the needs of its more technical users. This ongoing work is part of a broader trend where users adapt specialized hardware to their specific, often advanced, requirements, further solidifying the appeal of "hacker-friendly" devices like the reMarkable Paper Pro to a niche but influential segment of developers and power users. The journey towards a fully realized e-ink terminal on the Paper Pro underscores the enduring appeal of open platforms and the creative potential unleashed when users are empowered to explore and modify their technology.

Works cited

1. reMarkable Paper Pro | reMarkable, https://remarkable.com/store/remarkable-paper/pro?region_id=000840&gad_source=1 2. reMarkable Paper Pro, https://remarkable.com/store/remarkable-paper/pro 3. reMarkable Paper Pro Review: Is the New Color Tablet Worth the Upgrade? (2025), https://www.thequalityedit.com/articles/remarkable-paper-pro-review-vs-remarkable-2 4. Features and specifications - reMarkable Paper Pro | reMarkable, https://remarkable.com/store/remarkable-paper/pro/details/features 5. About reMarkable Paper Pro, https://support.remarkable.com/s/article/About-reMarkable-Paper-Pro 6. Software Stack - Developer Mode, https://developer.remarkable.com/documentation/software-stack 7. SDK documentation - Developer Mode, https://developer.remarkable.com/documentation/sdk 8. Developer mode for the reMarkable Paper Pro, https://support.remarkable.com/s/article/Developer-mode 9. Developer Mode - reMarkable, https://developer.remarkable.com/documentation/developer-mode 10. Remarkable Paper Pro – how to enable Developer Mode (and mini review) | IT Blog, https://www.informaticar.net/remarkable-paper-pro-how-to-enable-developer-mode-and-mini-review/ 11. Learn How to Access Your reMarkable Paper Pro Through the Command Line - Simply Kyra, https://www.simplykyra.com/blog/learn-how-to-access-your-remarkable-paper-pro-through-the-command-line/ 12. Switch Out Your reMarkable Paper Pro's Sleep Screen - Simply Kyra, https://www.simplykyra.com/blog/switch-out-your-remarkable-paper-pros-sleep-screen/ 13. SSH Access - reMarkable Guide, https://remarkable.guide/guide/access/ssh.html 14. Links - Developer Mode, https://developer.remarkable.com/links 15. reMarkable/linux-imx-rm: Linux kernel for reMarkable Paper Pro - GitHub, https://github.com/reMarkable/linux-imx-rm 16. Sam Littlefair: "This is a dream come true. I hacked my reMarkable e-ink tablet to run a terminal emulator. I'm now writing and coding in Helix Editor on my e-reader, and it feels amazing. First order of business: a five-star review of “The Bright Ages” by, https://bsky.app/profile/littlefair.ca/post/3llby7n73sc2n 17. Transferring Files - reMarkable Guide, https://remarkable.guide/guide/access/file-transfer.html 18. Getting Started — reMarkable Guide, https://remarkable.guide/ 19. reMarkable - Discord, https://discord.com/invite/rdFCveBJw7 20. Why Discord? : r/RemarkableTablet - Reddit, https://www.reddit.com/r/RemarkableTablet/comments/1gfz2ns/why_discord/ 21. Terminal (yaft) on my reMarkable : r/RemarkableTablet - Reddit,

https://www.reddit.com/r/RemarkableTablet/comments/1akkb3s/terminal_yaft_on_my_remarkable/ 22. reHackable/awesome-reMarkable: A curated list of projects related to the reMarkable tablet - GitHub, https://github.com/reHackable/awesome-reMarkable 23. danielebruneo/remarkable2-hacks: A collection of hacks, mods, tools, tips & tricks, specifically focused on the reMarkable 2 - GitHub, https://github.com/danielebruneo/remarkable2-hacks 24. What's the status of toltec development, and third-party apps on the Remarkable 2 in general? : r/RemarkableTablet - Reddit, https://www.reddit.com/r/RemarkableTablet/comments/16vm7j4/whats_the_status_of_toltec_development_and/ 25. toltec-dev/toltec: Community-maintained repository of free software for the reMarkable tablet. - GitHub, https://github.com/toltec-dev/toltec 26. Toltec for V3 : r/RemarkableTablet - Reddit, https://www.reddit.com/r/RemarkableTablet/comments/10awau6/toltec_for_v3/ 27. Discussions - toltec-dev toltec - GitHub, https://github.com/toltec-dev/toltec/discussions 28. timower/rM2-stuff: Collection of reMarkable related apps ... - GitHub, https://github.com/timower/rM2-stuff 29. Combine handwriting and typing with Type Folio for reMarkable, https://remarkable.com/using-remarkable/advanced-techniques/combine-handwriting-and-typing-with-type-folio 30. About Type Folio for reMarkable Paper Pro, https://support.remarkable.com/s/article/About-Type-Folio-for-reMarkable-Paper-Pro 31. davidegat/reMarkaDOS-Freedos-on-reMarkable: Simple FreeDOS setup, optimized for the reMarkable tablet, with CDROM support to mount own ISO image, and run old DOS software via BOCHS emulation. - GitHub, https://github.com/davidegat/reMarkaDOS-Freedos-on-reMarkable 32. Support for reMarkable Paper Pro · Issue #117 · owulveryck/goMarkableStream - GitHub, https://github.com/owulveryck/goMarkableStream/issues/117 33. Python libremarkable — Python libremarkable, https://libremarkable.eeems.codes/ 34. E Ink Gallery™ 3 | E Ink Brand | E Ink Product, https://www.eink.com/brand/detail/Gallery_3 35. Using the Linux FrameBuffer and libinput - Uno Platform, https://platform.uno/docs/articles/features/using-linux-framebuffer.html 36. Optimizing Linux desktop for Eink monitor : r/linuxquestions - Reddit, https://www.reddit.com/r/linuxquestions/comments/1hqjixx/optimizing_linux_desktop_for_eink_monitor/ 37. Can E Ink tablets really reduce stress? This study and my experience say yes - ZDNet, https://www.zdnet.com/article/can-e-ink-tablets-really-reduce-stress-this-study-and-my-experience-say-yes/ 38. It's 2025 and there is no Linux version of the reMarkable Desktop app - Reddit, https://www.reddit.com/r/RemarkableTablet/comments/1icq4b6/its_2025_and_there_is_no_linux_version_of_the/ 39. Ferrari to reveal six new cars in 2025 - one full electric - Granturismo Events, https://www.granturismoevents.com/story-ferrari-to-reveal-six-new-cars-in-2025-one-full-electric/ 40. Ferrari SF-25 development continues: major updates expected after Spanish technical directive - Scuderia Fans, https://scuderiafans.com/ferrari-sf-25-development-continues-major-updates-expected-after-spanish-technical-directive/