UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Porting di sistemi operativi Open Source su dispositivi mobile

Porting Open Source operating systems on mobile devices

Candidato:
Federico Paolucci

Relatore:
Chiar.mo Prof. Aldo Franco Dragoni

Anno Accademico 2021-2022

Indice

Elenco delle figure

Introduzione

1 Introduzione ai sistemi operativi mobile

Gli smartphone sono ormai diventati un elemento indispensabile nella vita quotidiana di molte persone grazie alle loro molteplici funzionalità e alla grande potenza di calcolo del loro hardware. Questi dispositivi, essenzialmente dei computer tascabili, per funzionare correttamente hanno bisogno di sistemi operativi appositi e progettati intorno ai loro componenti.
Ad oggi, chi detiene il monopolio sui sistemi operativi per cellulari sono Google con Android ed Apple con iOS. Secondo StatCounter, un servizio di analisi del traffico web, Android rappresenta circa il 72% del mercato degli smartphone mentre iOS detiene circa il 27% di market share: le due aziende californiane hanno concorrenza essenzialmente nulla in questo settore.
In passato hanno provato a ottenere una fetta di mercato anche Microsoft e Firefox con i loro rispettivi Windows Mobile e Firefox OS ma non sono riusciti a convincere il pubblico.
Chi, invece, sta convincendo gli appassionati è quell' 1% rimasto: sistemi operativi Open Source che mirano a portare l'esperienza Linux sui dispositivi mobile lasciando all'utente finale piena libertà di visionare tutto il codice e di modificarlo.
Il mercato dominato solo dai due principali attori Google ed Apple porta con sé diversi lati problematici che alcuni sviluppatori di progetti Open Source stanno cercando di contrastare con soluzioni interessanti. Queste soluzioni saranno esaminate e discusse nel corso della presente tesi mostrando sia gli aspetti positivi che quelli negativi dell'installazione di questi sistemi operativi.

2 Motivazioni e Obiettivi della tesi

La scelta di questo tema è motivata da diverse ragioni.
In primo luogo, è interessante poter effettuare un porting funzionante su ogni dispositivo mobile di un sistema operativo Open Source senza dipendere da Android e iOS. In secondo luogo, avere un'alternativa sul mercato significa anche poter scegliere di avere maggior privacy e sicurezza mentre utilizziamo il nostro smartphone.
Infine, l'elaborato può fornire informazioni utili creando una base dalla quale partire per eventuali sviluppi futuri, individuando aree di miglioramento e identificando opportunità di business nel mercato degli smartphone e dei sistemi operativi.

3 Struttura della tesi

La tesi è strutturata nel seguente modo:

- Il **Capitolo 1** si focalizza sulla situazione attuale del mercato dei cellulari, dominato da due soli sistemi operativi proprietari, Android di Google e iOS di Apple, analizzando il problema della gestione dei dati e della potenziale limitazione della privacy dell'utente.

- Il **Capitolo 2** esamina due progetti Open Source che intendono offrire un'alternativa, Ubuntu Touch e PostmarketOS, valutando la fattibilità di un porting su dispositivi già presenti sul mercato. Inoltre, vengono analizzate le proposte di alcune aziende lato hardware.

- Il **Capitolo 3** si concentra sullo sviluppo di un porting di Ubuntu Touch e PostmarketOS su un dispositivo specifico e vengono valutate le funzionalità disponibili e quelle mancanti rispetto al sistema operativo originale.

- Il **Capitolo 4** conclude la tesi con una valutazione dei risultati ottenuti e proponendo possibili sviluppi futuri per lo sviluppo del tema del porting di sistemi operativi mobile Open Source.

Capitolo 1

Mercato dei sistemi operativi mobile proprietari

1.1 Storia dei sistemi operativi mobile proprietari

Nei primi anni del 2000 il mercato dei sistemi operativi mobile non era ancora ben definito e i produttori di cellulari si concentravano soprattutto sulla funzione "telefono" lasciando spazio a poche altre funzionalità. Per ogni dispositivo veniva usato un sistema operativo specifico che, spesso, veniva in gran parte modificato per il modello successivo. L'evento che ha segnato un punto di svolta nel mercato dei dispositivi mobile è stata l'introduzione del primo iPhone da parte di Apple nel 2007. I cellulari, prima dotati di tasti fisici e di interfacce a menù complicati da navigare, ora presentavano un'interfaccia utente basata sul tocco semplificando l'utilizzo del dispositivo e rendendolo più facile da usare.
Sia Apple che Google lanciano sul mercato, per questa nuova tipologia di device, due sistemi operativi che sono diventati in poco tempo uno standard, spostando l'attenzione da chiamate e messaggi ad applicazioni ed accesso ad Internet. Vengono proposti, quindi, App Stores con i quali personalizzare il proprio telefono con le funzionalità che si desiderano e che consentono in caso di sostituzione di recuperare ciò che si era scaricato.
Android e iOS non hanno solo cambiato lo standard dei sistemi operativi per gli smartphone, ma hanno anche modificato quello di tablet e altri dispositivi multimediali come per esempio iPod e, successivamente, smartwatch. Grazie a questo cambiamento, il bacino di utenza e la fidelizzazione sono aumentati notevolmente

e chi intendeva acquistare un cellulare era sempre più indirizzato verso queste due soluzioni.

1.2 Principali attori: Android e iOS

Apple ha sviluppato iOS esclusivamente per i propri dispositivi, tra cui iPhone, iPad ed iPod Touch e ad oggi, nel mondo, circa 1,8 miliardi di questi sono attivi.
D'altra parte, Google ha rilasciato Android con l'obiettivo di renderlo disponibile per più device possibile non limitandosi solo agli smartphone ma estendendosi anche a smartwatch, tablet, TV e persino computer.
Tuttavia, come ha fatto quest'ultimo sistema operativo a diventare così popolare tra i produttori di hardware portatile? La risposta risiede nell'Android Open Source Project (AOSP). Android, infatti, è basato sul kernel Linux ed è distribuito sotto i termini della licenza libera Apache 2.0, pertanto gran parte del codice è Open Source e può essere visionato e modificato a proprio piacimento.
Android, però, non può essere considerato Open Source in toto: Google mantiene segreti alcuni aspetti chiave del sistema operativo come il processo di sviluppo dell'interfaccia utente e delle Google Apps, queste ultime di fondamentale importanza per il corretto utilizzo del software.
Ulteriore codice bloccato risiede nella parte di codice implementata dalle aziende produttrici di dispositivi mobile come Samsung, LG o Xiaomi tra le più conosciute, lasciando sostanzialmente ben poco di realmente "libero".

Android, comunque, rispetto al sistema operativo sviluppato da Apple, lascia maggior respiro agli utenti consentendo loro di apportare modifiche ad alcune sue funzionalità e permettendo l'installazione di applicazioni da fonti esterne al market-place. Fondamentalmente, il fatto che iOS ed Android siano vincolati a software proprietario implica che essi non siano da considerare sistemi operativi liberi ed Open Source, il che rende difficile conoscere il funzionamento interno dei due sistemi e le loro modalità di gestione dei dati.

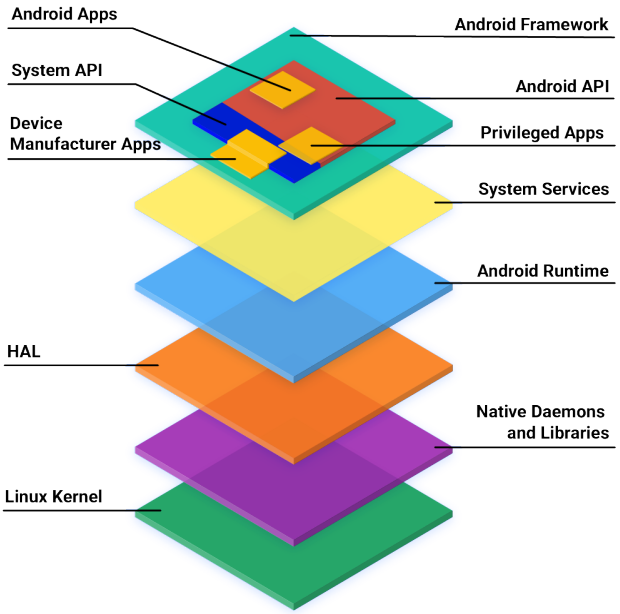

Figura 1.1: Stack Android: la piattaforma Android comprende sia componenti software Open Source che componenti proprietarie sviluppate da Google.

1.3 Limitazioni e problematiche

L'enorme predominanza di Google e Apple nel mercato dei sistemi operativi mobile è evidente, considerando che il 99% degli smartphone di tutto il mondo utilizza uno tra Android ed iOS. Ciò significa che gran parte del funzionamento di questi software è nascosto ai consumatori e che tutte le decisioni riguardanti le funzionalità, le politiche di sicurezza e la privacy sono prese esclusivamente da queste due società californiane, spesso con scarsa trasparenza e rendicontazione sulle loro scelte. Questo duopolio nei sistemi operativi mobile ha sollevato una serie di problemi e preoccupazioni sia per i fruitori che per le autorità governative che non hanno tardato a multare alcuni casi di violazione delle leggi antitrust.

In primo luogo, la problematica più evidente derivante da questo duopolio è la mancanza di scelta e di concorrenza: Google e Apple hanno una posizione di dominio assoluto che frena la libertà di scelta dei consumatori e costringe l'utente ad accettare le politiche e le funzionalità imposte dalle due aziende.

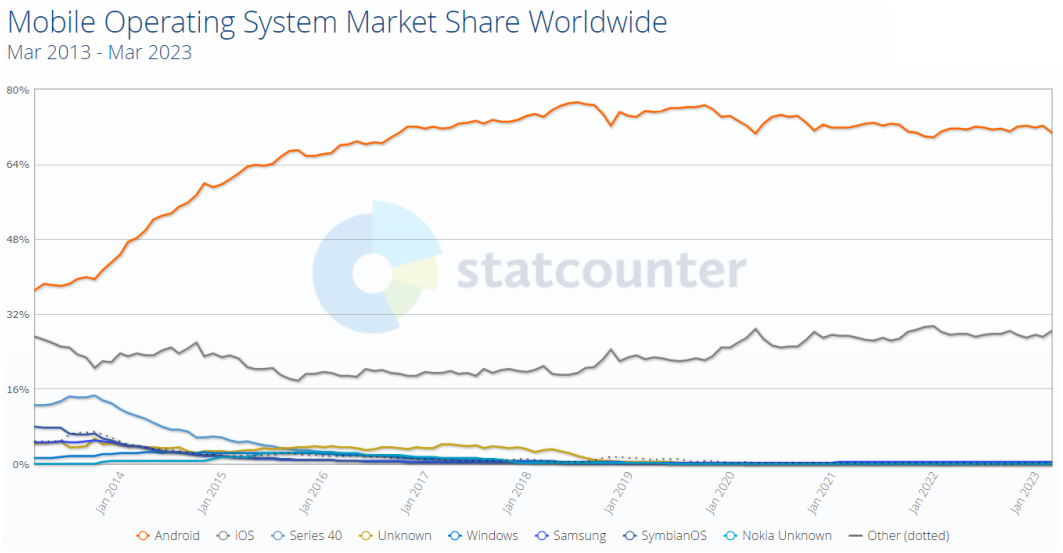

Figura 1.2: Market Share dei sistemi operativi mobile da marzo 2013 a marzo 2023

Questo rappresenta una limitazione anche per gli sviluppatori di applicazioni: un caso è quello della causa legale tra Epic Games ed Apple che ha visto la rimozione della loro applicazione dal marketplace a causa di politiche anti-steering (divieto a carico dello sviluppatore di dirottare l'utente verso servizi di pagamento esterni) troppo restrittive.
La scarsità di scelta si riflette, inoltre, anche sul costo dei prodotti in vendita che spesso sono più elevati nonostante la mancanza di innovazione nel settore.

La seconda problematica scaturita dal monopolio di Google ed Apple è quella riguardante il controllo dei dati e della privacy. Utilizzando lo smartphone, gli utenti forniscono alle due società una enorme mole di dati, tra cui informazioni ed interessi personali, cronologie di navigazione e posizioni GPS che vengono salvati, analizzati e rielaborati creando per ogni persona un profilo dettagliato. Tale profilo viene poi utilizzato per indirizzare pubblicità mirate e spingere gli utenti all'acquisto di determinati prodotti, oppure, dal punto di vista politico, per evidenziare determinate notizie o messaggi.
Il problema del controllo dei dati non si limita solamente a questo; i consumatori non hanno alcuna garanzia su dove vengano spedite le informazioni raccolte, in quanto il sistema operativo non fornisce la trasparenza del codice necessaria e questo comporta un pericoloso abuso della privacy.

I due colossi tecnologici hanno un enorme impatto sulla pubblicità online e chi utilizza i loro software è spesso indirizzato verso determinate pagine ed aziende: all'interno degli App Stores le applicazioni privilegiate sono quelle direttamente collegate con le due aziende, creando anche una situazione di svantaggio per la concorrenza.

Figura 1.3: Hub "Sicurezza e privacy" di Android 13 – Google ci assicura che il dispositivo sia protetto e aggiornato, ma la destinazione dei nostri dati è sconosciuta.

Un'ulteriore limitazione per gli utenti da non sottovalutare è quella della dipendenza dagli App Stores. I marketplace di Apple e Google sono il mezzo che permette agli sviluppatori di distribuire il loro prodotto al grande pubblico, rendendo il download e l'acquisto di un'applicazione molto semplice per gli utenti. Sebbene il controllo sui marketplace sia utile per prevenire l'installazione di software malevolo, esso rappresenta anche una riduzione della scelta delle applicazioni da scaricare. Android offre la possibilità di bypassare questa restrizione: abilitando l'installazione da fonti esterne all'App Store dalle impostazioni del device, tutti i file APK acquisiti dal Web o programmati dagli utenti possono essere correttamente montati.
iOS, invece, non offre alcuna opzione sotto questo punto di vista anche se, dal 2024, grazie al Digital Markets Act approvato dall'Unione Europea il 1° novembre 2022,

Apple sarà costretta a consentire all'installazione di app da fonti di terze parti.

Come ultimo punto, è importante sottolineare il problema della sicurezza. Molte risorse vengono impiegate per garantire la sicurezza delle varie versioni di Android e iOS, tuttavia, a causa del fatto che i sistemi operativi sono chiusi e proprietari, le vulnerabilità del sistema potrebbero rimanere nascoste a lungo. A causa di questo, la risoluzione di bug e falle nel software potrebbero richiedere diversi mesi prima di essere corretti mediante aggiornamenti OTA.
Per attenuare questi problemi, la soluzione che si andrà ad analizzare è quella dell'utilizzo di un dispositivo dotato di sistema operativo Open Source, lasciando maggior scelta e libertà ai consumatori, maggior trasparenza e controllo sui propri dati e fornendo un costante supporto grazie alle comunità di sviluppatori che continuamente migliorano e aggiornano il software.

Capitolo 2

Sistemi operativi mobile Open Source

Negli anni, a causa del monopolio di Google ed Apple nel mercato dei sistemi operativi mobile, alcuni sviluppatori hanno intrapreso l'iniziativa di fondare diversi progetti Open Source, che sono stati accolti dagli utenti in modo positivo. Alcuni di questi hanno avuto origine da una base Android, consentendo così il funzionamento dei loro sistemi operativi su dispositivi già in commercio, mentre altri sono partiti da Linux e lo hanno adattato per aggiungere funzionalità mobile.
Molti sono i tester che, desiderando un'alternativa, hanno installato sui loro cellulari i software proposti dalle comunità e nei forum dedicati si possono leggere prevalentemente impressioni e pensieri positivi, per questo motivo si può affermare che esiste una soluzione ai sistemi operativi proprietari e chiunque può già adottarla. In questo capitolo si analizzeranno i progetti Open Source più interessanti e di successo, con una particolare attenzione anche sugli smartphone che nascono abbracciando questo tipo di filosofia.

2.1 Filosofia Open Source

Il termine "Open Source" indica un tipo di licenza di distribuzione di software e applicazioni che consente lo studio, l'utilizzo, la modifica e la redistribuzione del codice sorgente.
L'approccio Open Source si basa sulla collaborazione e la partecipazione della comunità e consente di creare programmi di alta qualità e di risolvere problemi in modo più efficiente favorendo la flessibilità, consentendo agli utenti di modificare e personalizzare il software per adattarlo alle proprie esigenze.

Grazie alle "peer review", ovvero il processo di revisione collettiva del codice, gli sviluppatori correggono frequentemente eventuali bug e, a volte, vengono aggiunte funzionalità aggiuntive con la garanzia che il software sia costantemente migliorato per facilitarne l'utilizzo all'utente finale.
L'utilizzo di sistemi operativi Open Source può essere anche economico, poiché, spesso, disponibile gratuitamente o ad un costo molto ridotto rispetto a quelli proprietari. Ciò si ottiene senza compromessi e senza rinunciare alla qualità del prodotto.
Adottare una filosofia Open Source significa, quindi, offrire tutti questi benefici agli utenti contribuendo a creare un ambiente nel quale tutti possono trarre vantaggio da soluzioni software di alta qualità senza le restrizioni e le limitazioni tipiche dei sistemi operativi proprietari.

2.2 Ubuntu Touch

Nel 2013, Canonical presenta come risposta alla già famosa presenza di Android ed iOS nel settore, una versione ottimizzata per i dispositivi mobile di Ubuntu, denominata, appunto, Ubuntu Touch e nel 2015 l'azienda annuncia il lancio del primo smartphone ad implementarlo.
Il sistema operativo mantiene tutte le caratteristiche della controparte Desktop, perfino il layout è lo stesso con barra delle applicazioni laterali e barra di stato in alto. Viene distribuito con 12 applicazioni preinstallate come il lettore musicale, il calendario, la galleria, calcolatrice e persino il terminale; è possibile scaricare ulteriori applicazioni dallo store fornito da Canonical oppure eseguire nativamente quelle di Android mediante un layer di compatibilità chiamato Anbox.
Purtroppo, nonostante la promessa iniziale, Ubuntu Touch non riscuote molto successo e, nel 2017, Canonical smette di rilasciare nuove versioni del sistema operativo e chiude ufficialmente lo store.

In virtù della natura Open Source di Ubuntu Touch, il progetto non viene abbandonato ma viene subito preso in mano dalla UBPorts Foundation.
Fondata nel 2017 in Germania come fondazione di beneficienza, l'UBPorts Foundation si è posta l'obiettivo di supportare lo sviluppo collaborativo di Ubuntu Touch e

promuovere il suo utilizzo, fornendo supporto finanziario e legale alla comunità di developers.
Nel corso degli anni, l'organizzazione ha registrato un notevole aumento di dimensioni, dimostrando un impegno costante nei confronti del progetto e continuando a supportare i dispositivi che avevano già implementato il sistema operativo, oltre ad estendere la sua disponibilità ad un numero sempre crescente di smartphone.

Figura 2.1: Logo di UBPorts Foundation

Grazie all'aiuto di sviluppatori provenienti da tutto il mondo e all'appoggio economico degli sponsor, l'UBPorts Foundation è in procinto di rilasciare un nuovo major update che completerà la transizione di tutti i pacchetti del sistema da Ubuntu 16.04 LTS (edizione dell'ultima distribuzione utilizzata da Canonical) ad Ubuntu 20.04 LTS. Tale importante aggiornamento consentirà di modernizzare il software, che fino ad ora risultava piuttosto datato, rappresentando un notevole passo avanti per la piattaforma Ubuntu Touch.

Nativamente Ubuntu Touch viene reso disponibile sui dispositivi Volla Phone, tuttavia grazie agli sforzi della comunità, è possibile installarlo anche su altri cellulari. Al momento, tra i telefoni per i quali gli sviluppatori sono riusciti ad implementare più del 90% delle funzionalità originali troviamo: Google Pixel 3a/3a XL (con tutte le funzionalità perfettamente operative), Oneplus One, Google Nexus 5, Fairphone 2 e 4, vari prodotti Xiaomi e diversi altri, ciò a dimostrare la crescente compatibilità

del sistema operativo.

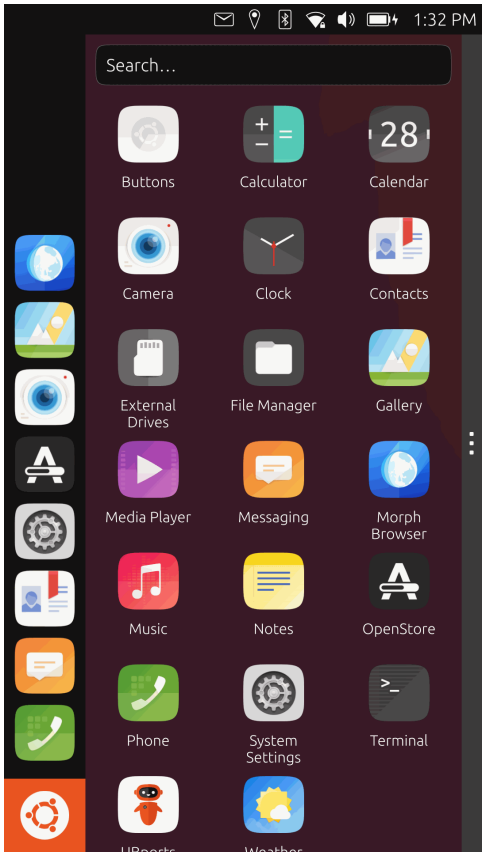

Figura 2.2: Uno screenshot del sistema operativo Ubuntu Touch

Il porting può essere effettuato anche su molti altri cellulari basati su Android sebbene la maggior parte di essi dipenda comunque da software proprietario. I produttori di smartphone, infatti, tendono a mantenere la parte di codice che comunica direttamente con l'hardware (driver di basso livello) proprietaria: queste componenti sono chiamate "vendor blobs" (Binary Large OBjects). Ubuntu Touch ha bisogno di questo tipo di software per implementare correttamente alcune importanti funzionalità come il modem, la wi-fi o la fotocamera.
Un altro elemento fondamentale per eseguire il porting di Ubuntu Touch è "Halium", una porzione di sistema operativo precompilata (filesystem di root) che contiene i file di base necessari per far funzionare il sistema operativo. Halium non comunica direttamente con l'hardware del dispositivo, ma richiede l'intermediazione dell'"Hardware Abstraction Layer" (HAL) che deve essere costruito su misura per ogni device in

quanto ognuno di essi presenta un'architettura hardware diversa. Sostanzialmente il progetto "Halium", anch'esso completamente Open Source, consente ai sistemi Linux di essere eseguiti su qualsiasi tipo di hardware Android.

Un porting di Ubuntu Touch ben riuscito apporta benefici anche alle prestazioni del dispositivo target, in quanto il sistema operativo tende ad essere più leggero e meno dispendioso in termini di risorse rispetto ad Android. Poiché Ubuntu Touch è progettato per essere fruibile anche su cellulari meno potenti, è ottimizzato per offrire un'interfaccia fluida e reattiva offrendo tempi di risposta rapidi.

In generale, Ubuntu Touch, rappresenta un'alternativa Open Source molto valida ad Android che potrebbe allungare di qualche anno la vita dei nostri device, qualora compatibili. In particolare, UBPorts Foundation ha previsto un supporto continuo con aggiornamenti per la sicurezza fino al 2030, dimostrando un impegno a lungo termine sul loro progetto.
Inoltre, la community attiva è fortemente interessata al testing e al miglioramento del sistema operativo contribuendo all'evoluzione di Ubuntu Touch.

2.3 PostmarketOS

PostmarketOS è un sistema operativo relativamente nuovo, il cui sviluppo è stato avviato nel 2017 da un gruppo di sviluppatori e appassionati di Linux. Questo progetto è nato per rispondere alla necessità di un sistema operativo Open Source leggero e adatto a una vasta gamma di dispositivi mobili, che spaziano dai cellulari e tablet Android, agli iPhone, alle console di gioco e anche ai dispositivi embedded.

Figura 2.3: Logo del sistema operativo PostmarketOS

L'obiettivo di PostmarketOS, così come dichiarato dal sito web ufficiale dell'organizzazione, è estendere la vita dei dispositivi, garantendo loro un supporto continuo anche dopo l'interruzione del supporto ufficiale, permettendo di mantenere i propri device aggiornati e utilizzabili per un periodo di tempo più lungo.
Anche in questo caso la natura Open Source del sistema operativo ha attirato l'interesse di numerosi appassionati e la community sta lavorando attivamente sul porting verso dispositivi non ancora compatibili. Tutto il codice sorgente del progetto è reso disponibile in modo trasparente sulla piattaforma GitLab, permettendo agli sviluppatori di contribuire, testare e migliorare il sistema operativo in maniera collaborativa.

PostmarketOS si basa su Alpine Linux, una distribuzione Linux leggera e molto efficiente caratterizzata da un basso consumo di risorse hardware e da dimensioni di download ridotte, appena 10 MB. Questa caratteristica consente uno sviluppo rapido e flessibile su qualsiasi distribuzione Linux, senza utilizzare macchine più potenti o molto spazio di archiviazione.

A differenza di Ubuntu Touch, che utilizza del codice proprietario, questo sistema operativo evita completamente l'ecosistema Android. Per questo motivo, piuttosto che costruire su misura l'immagine di sistema, l'intero OS è diviso in piccoli pacchetti indipendenti che possono essere installati su tutti i dispositivi che condividono la stessa architettura della CPU.

Questa modularità rende gran parte del sistema operativo standardizzato e utilizzabile su molteplici device semplificando di molto, non solo l'installazione, ma anche lo sviluppo e la manutenzione del software.

Idealmente, chiunque voglia iniziare ad utilizzare PostmarketOS dovrebbe avvalersi del Mainline Kernel ovvero un unico kernel ufficiale, compatibile con più dispositivi e più facile da tenere aggiornato. Tuttavia, spesso, per effettuare un porting bisogna partire dal kernel "downstream" personalizzato per l'hardware specifico su cui si lavora e poi cercare di sostituirlo con il Mainline Kernel, se possibile.

Gli utenti di PostmarketOS hanno la libertà di modificare qualsiasi aspetto del sistema operativo e, grazie alla sua modularità, questa operazione è semplificata: per applicare le modifiche effettuate, infatti, basta ricostruire il pacchetto.

Tuttavia, un aspetto negativo da tenere in considerazione se si desidera sperimentare PostmarketOS è la potenziale difficoltà nell'implementare alcune funzionalità cruciali come la connessione wi-fi o il modem per chiamate e messaggi, non integrando firmware proprietari.

Uno dei package più rilevanti che bisogna scegliere all'inizio dello sviluppo è quello della shell grafica: PostmarketOS implementa, ad oggi, 17 interfacce Open Source che spaziano da quelle simili a quelle utilizzate in ambiente desktop a quelle ottimizzate per dispositivi mobile.
Per i dispositivi che supportano il Mainline Kernel, ci sono tre interfacce grafiche ampiamente implementate:

- Phosh acronimo di Phone Shell, è basato su GNOME ed è stato sviluppato dall'azienda Purism per il loro smartphone. Grazie a PostmarketOS, può essere utilizzato anche su altri device. Phosh offre un layout molto simile a quello di Android e supporta un'interazione touch intuitiva, permettendo agli utenti di toccare, scorrere e pizzicare lo schermo per navigare tra le applicazioni, accedere alle funzionalità di sistema e interagire con il contenuto. Anche le notifiche sono supportate in modo efficace, garantendo una buona usabilità.

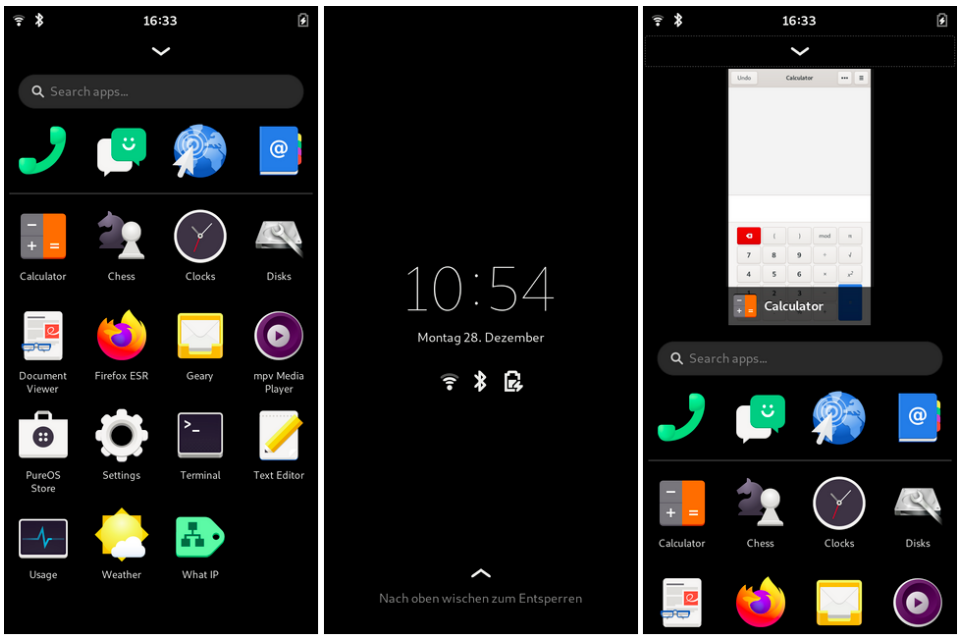

Figura 2.4: Esempio di interfaccia grafica Phosh

- Plasma Mobile, è la variante per dispositivi mobile e ottimizzata per il touch-screen, dell'ambiente grafico Plasma della comunità KDE. L'obiettivo principale di Plasma Mobile è quello di offrire una convergenza tra dispositivi mobile e

desktop, consentendo agli utenti di utilizzare lo smartphone come un computer tradizionale quando è collegato ad un monitor esterno. Questo ambiente pone, inoltre, una forte enfasi sulla sicurezza e sulla privacy degli utenti con funzionalità di crittografia dei dati, gestione dei permessi delle applicazioni e controllo dell'accesso alle risorse del device.

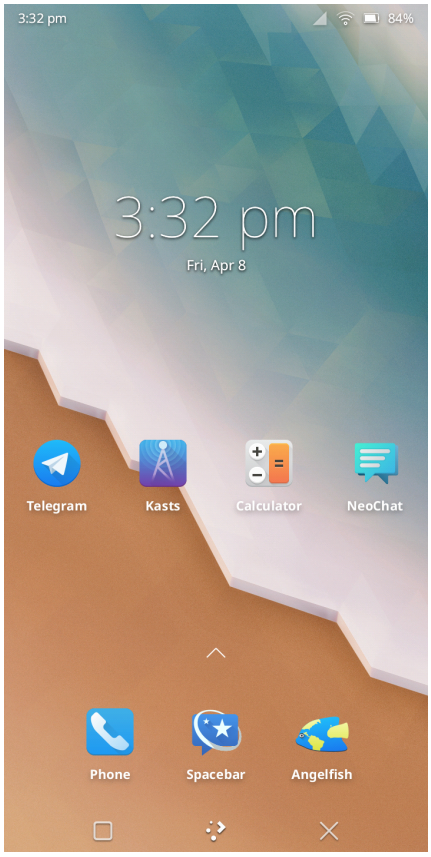

Figura 2.5: Esempio di interfaccia grafica Plasma Mobile

- Sxmo o Simple X Mobile, è un ambiente grafico leggero e minimalista progettato specificamente per dispositivi mobile e basato sul window manager dwm, noto per la sua efficienza e la sua filosofia di design semplice. Sxmo fornisce un'interfaccia utente snella e intuitiva che si adatta alle dimensioni dello schermo e alle risorse limitate dei cellulari, utilizzando il concetto di tilting window manager, dove le finestre vengono organizzate automaticamente senza sovrapporsi, ottimizzando l'utilizzo dello spazio a schermo. La scelta verso Sxmo, inoltre, può essere influenzata anche dal suo utilizzo della batteria,

essendo progettato per ridurre al minimo il consumo energetico dei dispositivi mobili, prolungandone l'autonomia.

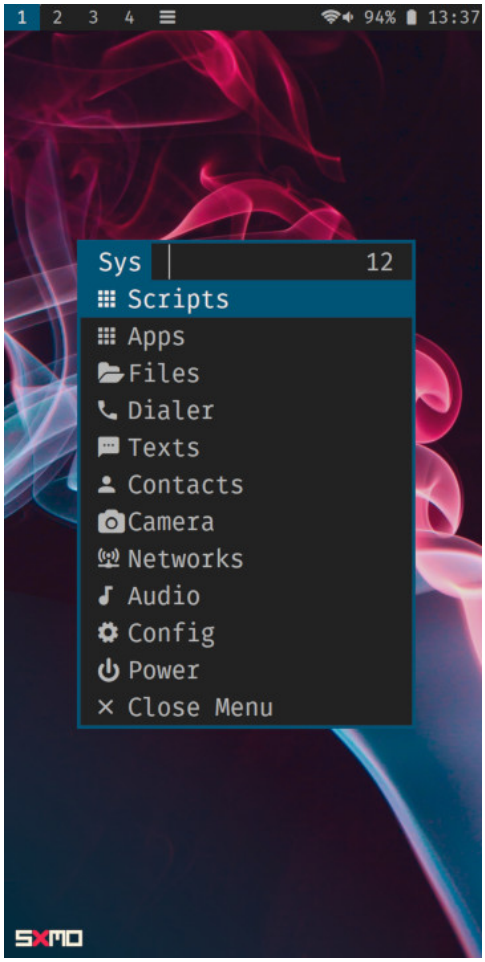

Figura 2.6: Esempio di interfaccia grafica Sxmo

PostmarketOS è un sistema operativo che, nonostante sia relativamente nuovo, ha già guadagnato una modesta, ma fedele, base di utenti e sviluppatori. Sebbene sia ancora un lungo cammino da percorrere per raggiungere il livello di immediatezza e popolarità di Android e iOS, ci sono segnali positivi che indicano un crescente interesse verso il progetto.

2.4 Hardware "Open Source"

Un utente meno esperto in termini di porting e di installazioni personalizzate sui propri dispositivi non è escluso dall'uso di un sistema operativo Open Source. Alcune aziende hanno risposto alle esigenze dei consumatori proponendo nel mercato hardware, degli smartphone che sono aperti a tutti i tipi di progetti Open Source come quelli menzionati in precedenza. Queste soluzioni consentono agli utenti di beneficiare dei vantaggi offerti dal software Open Source, senza dover necessariamente possedere esperienze tecniche avanzate e, spesso, ad un prezzo inferiore rispetto ai più famosi device Android e iOS.
Questo scenario sta aprendo la strada ad un uso più diffuso dei sistemi operativi aperti e sta cercando di aumentare la quota di mercato di tali sistemi, rendendoli noti al grande pubblico.
Tra le aziende che offrono questo tipo di soluzione, spiccano Pine64, Purism e Fairphone.

2.4.1 PinePhone

Pine64 è una società che si autodefinisce "community-driven", ovvero basata sulla sua attiva comunità di utenti, e si impegna a produrre dispositivi di alta qualità e a basso costo basati sull'architettura di processori ARM.
Offrono una vasta gamma di device che integrano sistemi operativi Open Source, tra cui telefoni, tablet, smartwatch, portatili e single-board computers utilizzati per server o automazioni industriali.
Il PinePhone è la proposta di smartphone dell'azienda, in grado di eseguire versioni "Mainline" del kernel Linux senza bisogno di adattamenti personalizzati come accade per i telefoni Android. Questa caratteristica lo rende particolarmente interessante per i sistemi operativi Open Source menzionati in precedenza: tutte le funzionalità possono essere implementate dagli sviluppatori senza problemi di compatibilità con firmware proprietari.
La prima versione disponibile del PinePhone, chiamata Braveheart Edition, veniva spedita senza un sistema operativo preinstallato offrendo la possibilità, in particolare agli utenti esperti interessati al progetto, di implementare quello che preferivano. Le

versioni successive, invece, sono state rese più accessibili per tutti gli altri utenti, utilizzando l'interfaccia Plasma Mobile sul sistema operativo Open Source Manjaro Linux.
Un aspetto unico del PinePhone è la possibilità di poter effettuare il boot selezionando un sistema operativo da una microSD rimovibile, distinguendolo dagli altri dispositivi mobile.
Per coloro che desiderano modificare e personalizzare il software del telefono, sono disponibili sulla wiki ufficiale tutti i datasheet delle componenti hardware e, inoltre, è possibile disattivare alcune di queste attraverso degli switch fisici posizionati sotto la scocca sul retro.

Figura 2.7: Retro del Pinephone con kill switch fisici per disabilitare alcune delle componenti hardware

In generale, PinePhone concede una reale piena libertà all'utente di fare ciò che desidera sul dispositivo: il software è completamente libero da firmware proprietari e perfino l'hardware può essere sostituito, in base alle esigenze, con altri componenti compatibili.

2.4.2 Librem 5

Un'altra azienda che si è fatta portavoce dei principi dell'Open Source, della libertà e della privacy e si schiera apertamente contro Google e Apple, è la californiana Purism con i suoi Laptop Librem e lo smartphone Librem 5.
Purism gestisce lo sviluppo del sistema operativo PureOS, una distribuzione di Linux gratuita basata su Debian che non include i software della repository Debian che

violano le “GNU Free System Distribution Guidelines”, linee guida che qualificano quando un software è “libero”.

Nel dicembre 2017, proprio per questo motivo, il sistema operativo è stato approvato dalla Free Software Foundation.

PureOS è il sistema operativo scelto per i loro prodotti, in particolare per il loro cellulare Librem 5, dove gli utenti hanno comunque la possibilità di installare qualsiasi alternativa: “It’s your hardware” citando il sito ufficiale.

Il Librem 5 è progettato concentrandosi su sicurezza e protezione della privacy e Purism ha specificato che il suo smartphone sarebbe diventato il primo dispositivo mobile IP-native ad utilizzare una comunicazione end-to-end criptata e decentralizzata. Ciò significa che il telefono è in grado di gestire la comunicazione diretta tra due dispositivi utilizzando il protocollo internet (IP) senza passare attraverso un server centralizzato controllato da un’altra azienda, ma navigando in una rete di nodi distribuiti in tutto il mondo. Inoltre, grazie alla crittografia end-to-end, solo i destinatari autorizzati possono decodificare i dati trasmessi.

Un’altra caratteristica degna di nota sono i kill switch fisici per disattivare Wi-Fi, Bluetooth, fotocamera, microfono e processore di baseband (parte del modem che gestisce le comunicazioni cellulari). Quest’ultimo, inoltre, è separato dalla CPU impedendo l’accesso indesiderato a dati sensibili come la rubrica, i messaggi e le informazioni di localizzazione.

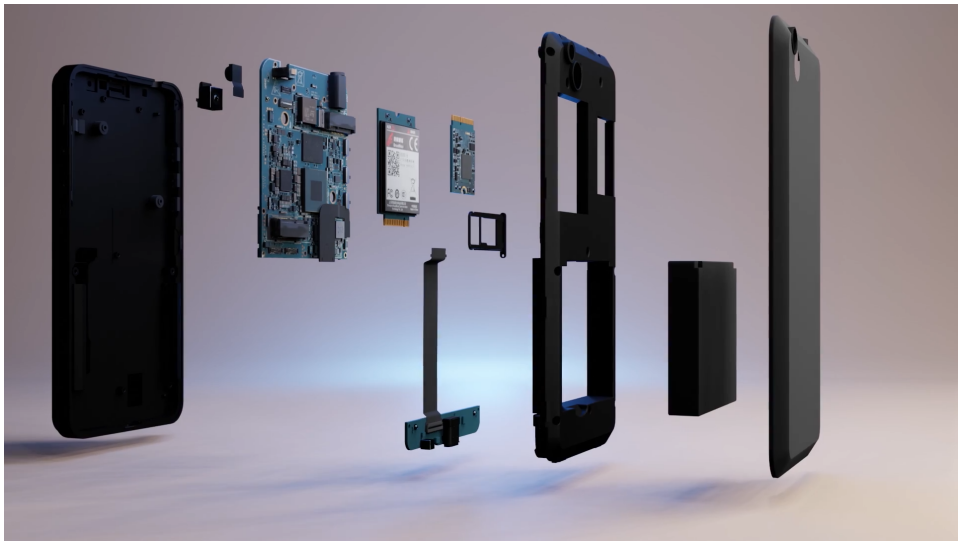

Figura 2.8: Design modulare del Librem 5

Tutti questi elementi rendono il Librem 5 molto diverso da quanto proposto sul mercato e consentono agli utenti di avere pieno controllo sulla propria privacy anche senza avere conoscenze tecniche avanzate in materia.
Raccogliendo circa 2 milioni di dollari in due mesi per il crowdfunding del progetto, Purism ha ottenuto grande interesse da parte del pubblico. L'azienda ha una visione chiara sul futuro dei suoi smartphone, il che lo rende un'ottima scelta per i consumatori più esigenti.

2.4.3 Fairphone

Un'altra soluzione disponibile è quella proposta da Faiphone, un'azienda nata nei Paesi Bassi nel gennaio 2013 che punta ad una direzione diversa rispetto alle società precedentemente introdotte. Fairphone propone uno smartphone prodotto in condizioni di equità, garantendo che i minerali utilizzati non provengano da miniere controllate dai signori della guerra civile del Congo, e che i lavoratori coinvolti nell'assemblaggio godano di diritti sindacali.

Figura 2.9: Fairphone 4, l'ultimo modello rilasciato dall'azienda

Per quanto riguarda il software, Fairphone inizialmente utilizzava Android come sistema operativo predefinito ma, a partire dalla versione 2 del cellulare è possibile scegliere tra diverse opzioni, tra cui Ubuntu Touch, Sailfish OS e LineageOS (una versione modificata di Android).
È importante notare che, a differenza di PinePhone e Librem 5, questo telefono è il

“meno open” in termini di personalizzazione, infatti, come spiegato dai manutentori del progetto Replicant, la scelta di utilizzare processori Qualcomm non consente l’installazione di sistemi operativi senza utilizzare anche firmware proprietari.
Nonostante le limitazioni, il progetto Fairphone ha ottenuto un apprezzabile successo e con le dovute correzioni, potrebbe rappresentare una rampa di lancio per un mercato più libero in futuro.

Capitolo 3

Analisi del porting di un sistema operativo mobile Open Source

Un utente potrebbe decidere di non acquistare un nuovo dispositivo al solo scopo di sperimentare l'esperienza Open Source, ma potrebbe invece optare per eseguire il porting di un sistema operativo mobile libero su uno dei suoi dispositivi. Tuttavia, poichè quelle da effettuare sono operazioni delicate, è generalmente raccomandato di evitare l'utilizzo di un cellulare impiegato quotidianamente, poiché esiste la possibilità, per quanto rara, di incorrere in errori irreversibili come ad esempio l'hardbrick del telefono.
In questo capitolo, si analizzerà la fattibilità del porting di Ubuntu Touch e PostmarketOS su un dispositivo Android.
Per la costruzione delle immagini di sistema e il successivo flash sul telefono, è stato utilizzato un computer dotato della macchina virtuale VirtualBox con il sistema operativo Ubuntu 22.04 installato.

3.1 Dispositivo di destinazione

Il dispositivo specifico su cui verrà eseguito il porting nelle fasi successive è il "A5 2016" prodotto da Samsung e comunemente noto con il nome in codice "a5xelte". Essendo un prodotto Samsung, utilizza dei blobs proprietari che possono essere ottenuti da un repository su Github.
Il kernel da modificare, anch'esso reperibile da Github grazie allo sforzo di alcuni utenti, è un kernel universale per il SoC Samsung Exynos 7580 che è condiviso da

altri dispositivi della stessa "famiglia".
Prima di procedere con il porting, sono state eseguite alcune operazioni preliminari sul dispositivo. Queste includono lo sblocco del bootloader e l'installazione di una recovery modificata: quest'ultima modifica il software Knox, un sistema di controllo che tiene traccia delle modifiche apportate al sistema operativo, invalidando la garanzia del cellulare.

Il bootloader è un programma che viene eseguito durante l'avvio del dispositivo, responsabile del caricamento del sistema operativo che di solito è bloccato dal produttore per garantirne la sicurezza ed impedire l'installazione di software non autorizzato. Tuttavia, è possibile eseguire una procedura di sblocco per bypassare questa protezione. Questa procedura è generalmente semplice: dalle impostazioni si abilita e si accede al menù "Opzioni sviluppatore" e si spunta l'opzione apposita. I passaggi specifici possono variare leggermente a seconda dello smartphone e della versione di Android utilizzata.

La recovery è una particolare modalità di avvio del sistema operativo, attivabile in fase di accensione tenendo premuto il pulsante del volume su e il pulsante fisico centrale. Essa permette di eseguire diverse operazioni, come il ripristino dei dati di fabbrica o l'installazione di un aggiornamento tramite una scheda SD esterna o mediante il protocollo ADB.
L'installazione di una recovery modificata fornisce nuove funzionalità che saranno utili durante le fasi di porting tra cui l'installazione di immagini di sistema, la formattazione di varie partizioni del disco, l'accesso al terminale e la possibilità di trasferire file tramite ADB da e verso qualunque cartella del device.
Nel caso specifico, è stata scelta l'installazione di TWRP (Team Win Recovery Project), che è considerata la più completa e supportata tra le custom recovery Open Source.

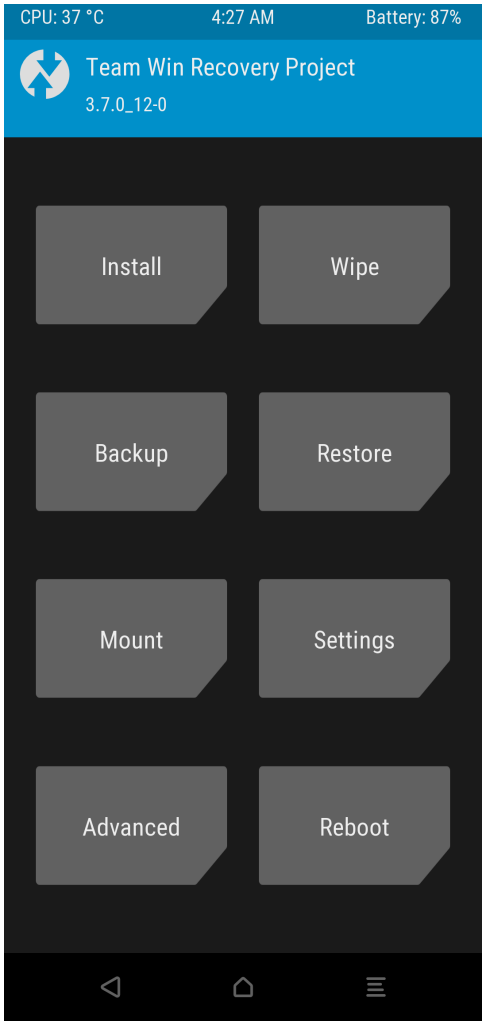

Figura 3.1: Menù con tutte le funzionalità della recovery TWRP

Dopo aver completato con successo queste operazioni preliminari è possibile procedere con il processo di porting e la costruzione delle immagini da installare sul dispositivo.

3.2 Porting di Ubuntu Touch

Per eseguire il porting di Ubuntu Touch sono necessari tre componenti fondamentali:

- Il root filesystem di Ubuntu Touch o rootfs: si tratta dell'immagine di sistema che deve essere avviata al momento del boot del dispositivo. All'interno del rootfs sono presenti i file e le directory base del sistema operativo.
- Il framework Halium correttamente configurato per il dispositivo usato.

- I blobs proprietari, in questo caso quelli forniti dal produttore Samsung che includono driver e firmware specifici per il a5xelte.

Il primo passo intrapreso è stato la compilazione di Halium per il dispositivo a5xelte. Poiché viene costruito usando il codice sorgente di una versione modificata di Android chiamata LineageOS, è necessario individuare la versione corretta di Halium da usare in base alla versione di Android del cellulare.
Nel caso specifico, è stata utilizzata la versione di Halium 9.0 creata partendo da LineageOS 16.0 (versione Android 9.0).
Proprio da questa versione è stato introdotto un cambiamento per quanto riguarda i blobs proprietari: il codice non condivide più la stessa partizione del resto dell'immagine di sistema, ma risiede in una partizione separata.
L'immagine di sistema senza blobs, rinominata Generic System Image (GSI), semplifica il processo di "building" del porting, rimuovendo la necessità di adattamento, compilazione e configurazione dei blobs nell'immagine.
Con questa versione, sono possibili tre approcci differenti:

- Metodo Full system image: Utilizzato per versioni precedenti di Halium (fino alla 7.1), coinvolge la compilazione sia dell'immagine di boot (halium-boot.img) che della specifica immagine di sistema (system.img) dal codice sorgente di LineageOS, seguite dall'installazione congiunta con il file system di root fornito da UBPorts (rootfs).

- Metodo Halium-boot: comporta la compilazione di halium-boot.img e la sua installazione insieme alla Generic System Image di Halium e al rootfs di UBports. Questo metodo è quello scelto per il dispositivo a5xelte ed è applicabile solo per le versioni di Halium dalla 9.0 in poi, sfruttando proprio il cambiamento architetturale introdotto da Android 9.0.

- Metodo Standalone kernel: Questo metodo richiede solo la compilazione del kernel e l'installazione della GSI insieme al ramdisk di Halium e il rootfs di UBports. Anche questo metodo è applicabile solo alle versioni successive a Halium 9.0. La componente aggiuntiva da installare è il ramdisk, un'immagine temporanea del file system caricata nella RAM durante il processo di avvio, che fornisce un'interfaccia comune per il kernel Linux e il sistema Android

permettendo di utilizzare le funzionalità hardware del dispositivo anche su diverse distribuzioni Linux.

Nel caso specifico, si utilizzerà il metodo Halium-boot.

3.2.1 Configurazione ambiente di sviluppo

Per configurare l'ambiente di sviluppo su cui verrà costruita l'immagine, è necessario eseguire alcuni passaggi. Innanzitutto, su una macchina host con Ubuntu 22.04, è importante abilitare l'architettura i386 per garantire la compatibilità dei pacchetti. Successivamente, ho installato le dipendenze di cui avevo bisogno con il comando "sudo apt install".

```
sudo apt install git gnupg flex bison gperf build-essential \
zip bzr curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
libgl1-mesa-dev g++-multilib mingw-w64-i686-dev tofrodos \
python3-markdown libxml2-utils xsltproc zlib1g-dev:i386 schedtool \
liblz4-tool bc lzop imagemagick libncurses5 rsync \
python-is-python3 python2
```

Figura 3.2: Lista delle dipendenze installate per configurare l'ambiente di sviluppo

Questi pacchetti sono necessari per assicurare il corretto funzionamento degli strumenti che verranno utilizzati durante lo sviluppo.
Infine, ho scaricato e reso eseguibile lo script "repo" utile per gestire il codice sorgente da utilizzare. Tra le sue molte funzioni, una di queste permette di clonare repository Git in cartelle locali facilitando il lavoro su di esse.

3.2.2 Configurazione Halium

È il momento di scaricare, all'interno del computer configurato, il codice sorgente di Halium.
Ho creato una directory denominata "halium" e con il comando: "repo init -u https://github.com/Halium/android -b halium-9.0 –depth=1" ho selezionato e inizializzato la corretta versione da clonare da GitHub, successivamente con "repo sync" ho avviato il download.
Il passo successivo è quello di integrare il codice scaricato con il codice specifico del dispositivo a5xelte, che è necessario per costruire il kernel e per far funzionare

correttamente l'hardware del telefono.
Per fare ciò, bisogna creare un file .xml chiamato "manifest" in cui verranno specificati i repository online da dove reperire quello che ci serve. Il manifest del dispositivo deve essere inserito nella directory "halium/halium/devices/manifests" e rinominato "samsung_a5xelte.xml" (la prima parte del nome indica il produttore e la seconda parte indica il nome dello smartphone).
Prima di aggiungere contenuto al file, è importante localizzare tutti i repository necessari. Non è sufficiente includere il solo repository del device con all'interno i suoi driver, ma abbiamo bisogno anche di tutti i file da cui dipende. Questi possono essere individuati grazie al file lineage.dependencies che contiene tutti i repository nei quali sono inseriti.

Dopo aver ottenuto queste informazioni si può modificare il file manifest. Le righe di codice inserite nel tag radice <manifest> sono così strutturate:
<project path="..." name="..." remote="..."/>
Il tag "project" contiene tutti gli attributi necessari a recuperare il repository remoto.

- L'attributo "path" specifica il percorso in cui salvare il codice;
- L'attributo "name" specifica il nome del repository remoto
- L'attributo "remote" specifica da dove reperire il repository. Questo attributo in particolare ha bisogno di alcune configurazioni aggiuntive approfondite in seguito.

Nel caso del dispositivo a5xelte la lista completa dei tag "project" è:

```
<project path="device/samsung/a5xelte"
name="android_device_samsung_a5xelte" remote="device"/>
<project path="device/samsung/universal7580-common"
name="android_device_samsung_universal7580-common" remote="device"/>
<project path="hardware/samsung"
name="android_hardware_samsung" remote="device"/>
<project path="packages/resources/devicesettings"
name="android_packages_resources_devicesettings" remote="device"/>
```

```
<project path="hardware/samsung_slsi/exynos"
name="android_hardware_samsung_slsi_exynos" remote="device"/>
<project path="hardware/samsung_slsi/exynos5"
name="android_hardware_samsung_slsi_exynos5" remote="device"/>
<project path="hardware/samsung_slsi/exynos7580"
name="android_hardware_samsung_slsi_exynos7580" remote="device"/>
<project path="hardware/samsung_slsi/openmax"
name="android_hardware_samsung_slsi_openmax" remote="device"/>
<project path="kernel/samsung/universal7580"
name="android_kernel_samsung_universal7580" remote="device"/>
<project path="vendor/samsung" name="proprietary_vendor_samsung"
remote="blobs"/>
```

Si può notare che molti dei repository sono comuni con altri dispositivi che utilizzano il processore Exynos 7580.

Per quanto riguarda l'attributo "remote", questo deve essere configurato nel tag "remote" nel seguente modo: <remote name="..." fetch="..." review="..." revision"..."/>

- L'attributo "name" è il nome da scrivere sull'attributo "remote" del tag "project" per fare riferimento allo specifico remote configurato;
- L'attributo "fetch" indica l'URL da cui il codice del repository remoto può essere scaricato;
- L'attributo "review" indica l'URL della piattaforma di code review utilizzata per il repository remoto, questo attributo è spesso omesso;
- L'attributo "revision" indica la specifica revisione o branch del repository da utilizzare.

Nel caso del dispositivo a5xelte sono stati inseriti:

```
<remote name="device" fetch="https://github.com/LineageOS"
revision="lineage-16.0" />
```

```
<remote name="blobs" fetch="https://github.com/TheMuppets"
revision="lineage-16.0" />
```

Concludendo con </manifest>, viene completato il file manifest per lo smartphone a5xelte.

Ora si possono sincronizzare i repository nella directory locale. Per fare ciò, bisogna avviare lo script "setup" contenuto nel percorso "halium/halium/devices", passando come parametro il nome del dispositivo.
Finito il processo, si potrebbe eseguire lo script "apply-patches.sh" per applicare alcune patch utili ad evitare errori durante il processo di build dell'immagine, ma, dato che ho scelto il metodo halium-boot si può omettere questo passaggio.

Il passo successivo consiste nell'impostare le variabili di ambiente necessarie per eseguire alcuni comandi di compilazione del codice di Android.
Per fare ciò, viene eseguito, nel terminale aperto nella cartella "halium" creata in precedenza, il comando "source build/envsetup.sh" che, inoltre, indica alla shell dove trovare gli strumenti di sviluppo di Android e dove archiviare i risultati della compilazione.
Ora si può procedere, eseguendo il comando "breakfast" seguito dal nome del cellulare utilizzato, a5xelte, impostando l'ambiente di compilazione per il dispositivo specificato.

3.2.3 Personalizzazione kernel

È opportuno, adesso, personalizzare il kernel, poichè quello scaricato tramite il manifest, è impostato per LineageOS, una versione modificata di Android.
Per fare ciò, viene utilizzato un altro script denominato "check-kernel-config" che si trova in "halium/halium/halium-boot" che individua la configurazione adatta ad Ubuntu Touch passando come parametro il percorso del file di configurazione denominato "lineageos_a5xelte_defconfig" e il flag "-w" per indicare al programma di scrivere eventuali avvertimenti generati durante il controllo in un file log. La

posizione del defconfig, nella directory kernel, dipende dall'architettura del device che si sta usando: arch/arm/configs se è a 32bit, arch/arm64/configs se è a 64bit.

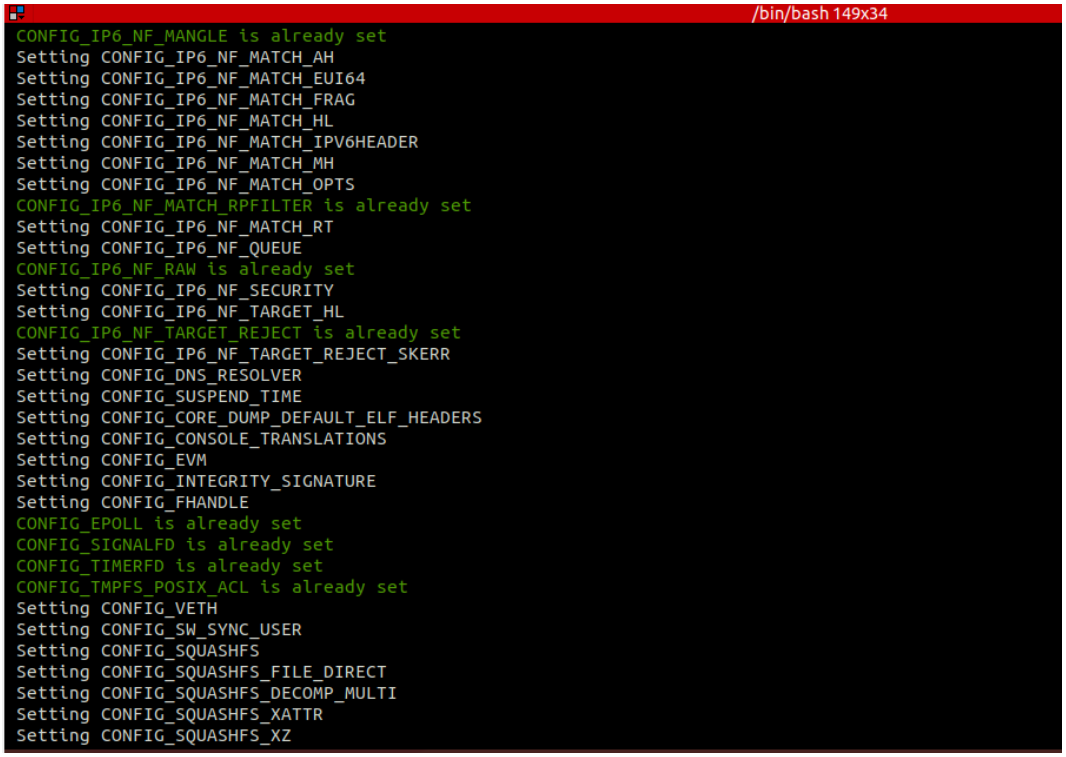

Figura 3.3: Esempio di output dello script "check-kernel-config"

Non tutto viene configurato correttamente tramite questo script, pertanto è necessario aprire il file defconfig e impostare manualmente alcuni parametri come indicato nel log prodotto, in particolare "CONFIG_CMDLINE="console=tty0"" e "CONFIG_CMDLINE_EXTEND=y". Questo permetterà l'accesso al dispositivo tramite protocollo SSH dopo l'avvio del sistema operativo.
Inoltre, per Halium versione 9, bisogna costruire l'immagine di sistema come "system-as-root". In pratica, la partizione "system" diventerà la radice del filesystem e tutte le altre partizioni ("data", "cache"...) sono montate come sottodirectory di "system". Ho quindi aggiunto al file "boardconfig.mk" situato nella directory "halium/device/samsung/a5xelte":

```
BOARD\_BUILD\_SYSTEM\_ROOT\_IMAGE := true
```

Dato che questa modifica renderà la partizione root in Android di sola lettura per motivi di sicurezza e stabilità, ho specificato una lista di punti di montaggio aggiuntivi per alcune partizioni che devono essere accessibili:

```
BOARD_ROOT_EXTRA_FOLDERS := \
/firmware \
/disp \
/persist
```

3.2.4 Compilazione "halium-boot.img"

Lo step conclusivo consiste nella creazione dell'immagine di boot. Halium usa il tool "mkbootimg" che consente di creare un'immagine personalizzata per il dispositivo Android, combinando il kernel, il ramdisk e le informazioni sulla configurazione del bootloader.
Per avviare il processo si esegue il comando "mka mkbootimg" nel terminale. Si completa definitivamente l'operazione e si costruisce "halium-boot.img" con i seguenti comandi:

- "export USE_HOST_LEX=yes" per impostare l'utilizzo di una libreria di analisi lessicale ospite durante la compilazione;
- "mka halium-boot" per avviare il processo di compilazione.

Ho riscontrato, durante l'esecuzione di "mka halium-boot", un errore di incompatibilità tra tipi (gid_t dovrebbe essere kgid_t oppure uid_t dovrebbe essere kuid_t) che ho risolto modificando e disabilitando il parametro "CONFIG_USER_NS". Se avessi scelto il metodo completo di "system.img" avrei dovuto creare anche un'altra immagine, ma, per questo caso specifico ho evitato questo passaggio.

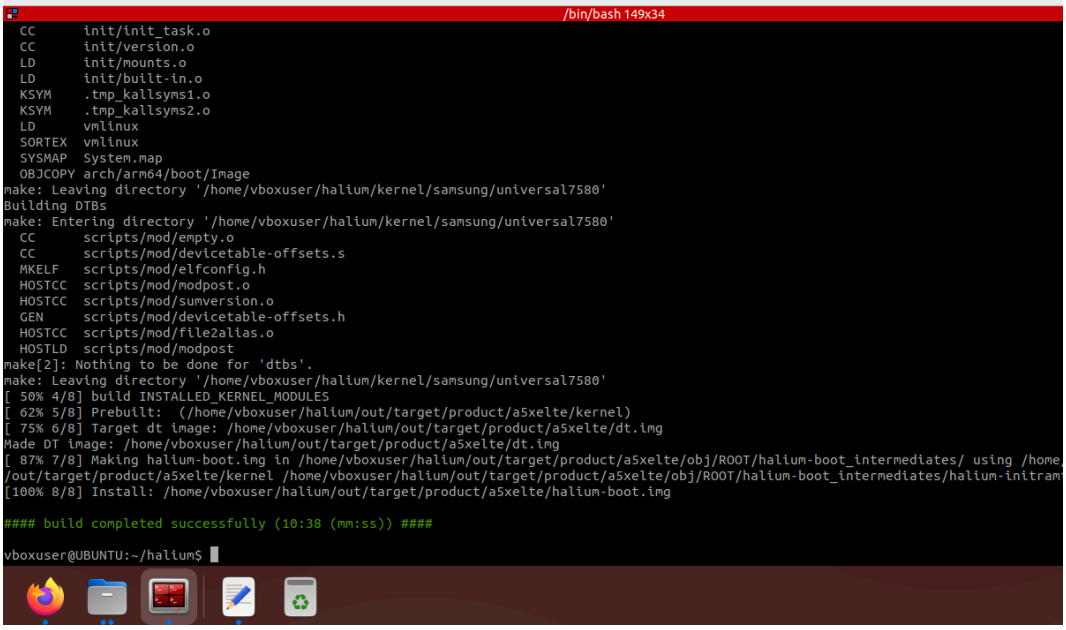

Figura 3.4: Compilazione dell'immagine "halium-boot.img" completata con successo

3.3 Installazione Ubuntu Touch e problematiche

Per l'installazione di Ubuntu Touch, è necessario utilizzare la recovery TWRP o qualsiasi altra recovery che faccia uso di Busybox, un insieme di strumenti aggiuntivi alla recovery di default. Avviando il cellulare in modalità recovery, bisogna formattare la partizione /data in ext4 e rimuovere la crittografia.
Per scrivere il file "halium-boot.img" nella partizione "boot" del dispositivo esistono diversi metodi:

- Per gli smartphone che consentono l'avvio in modalità fastboot (protocollo di avvio rapido), bisogna aprire il terminale nella directory dove è stata costruita l'immagine e, dopo aver connesso al PC il cellulare in modalità fastboot, digitare il comando "fastboot flash boot halium-boot.img".

- Per gli smartphone che consentono l'avvio in modalità download, ad esempio dispositivi Samsung come il modello a5xelte, è necessario utilizzare il software Heimdall se si utilizza Linux sul computer host, oppure il software Odin se si utilizza Windows. L'interfaccia dei due programmi è molto intuitiva: è sufficiente selezionare il file da installare e far partire il processo. Al termine il dispositivo si riavvierà automaticamente.

- In caso di problemi con i due metodi precedenti, si può installare il file tramite la modalità recovery. Utilizzando il protocollo adb, si sposta il file sul dispositivo con il comando "adb push halium-boot.img /tmp/" poi, si seleziona l'immagine dal menù "Install" di TWRP e si indica la partizione di destinazione: in questo caso "Boot".

Per completare l'installazione di Ubuntu Touch bisogna aver scaricato le rootfs adeguate alla versione di Halium e all'architettura del dispositivo, insieme alle GSI. Si riavvia il cellulare in modalità recovery e ci si assicura che il dispositivo sia rilevato dal computer attraverso il protocollo adb, poi, si procede con il cloning del repository e l'installazione sul device con lo script:
"halium-install -p ut16.04 rootfs.tar.gz system.img".
In questo comando, "ut16.04" è la versione di Ubuntu che si sta installando, mentre "system.img" è il file estratto dalle GSI.
Durante il processo, tutti i file necessari verranno collocati nelle rispettive directory adeguate e prima di terminare l'operazione verrà chiesto di creare una password per l'accesso al dispositivo.

Purtroppo, con il device a5xelte non è stato possibile avere un boot riuscito a causa di alcuni errori che si sono presentati in fase di debug.
Il termine "boot riuscito" si riferisce alla possibilità di stabilire una comunicazione con il dispositivo, che ha correttamente avviato Ubuntu Touch, attraverso il protocollo ssh per la procedura di debug e configurazione, anche senza un'interfaccia grafica non ancora correttamente impostata.
In questo caso specifico sono riuscito a comunicare solo attraverso una connessione telnet al fine di recuperare alcuni log del kernel e tentare di risolvere il problema. Dopo aver stabilito la connessione telnet con "telnet 192.168.2.15" nel terminale del computer host, il primo log recuperato tramite il comando "dmsg" indicava che la console non era stata inizializzata, causando l'interruzione del processo di boot in Ubuntu Touch. Quindi, ho verificato che tutti i parametri di configurazione del kernel riguardanti la console fossero correttamente impostati e ho notato che CONFIG_VT era disabilitato, nonostante fosse indicato come abilitato nel file di configurazione defconfig.

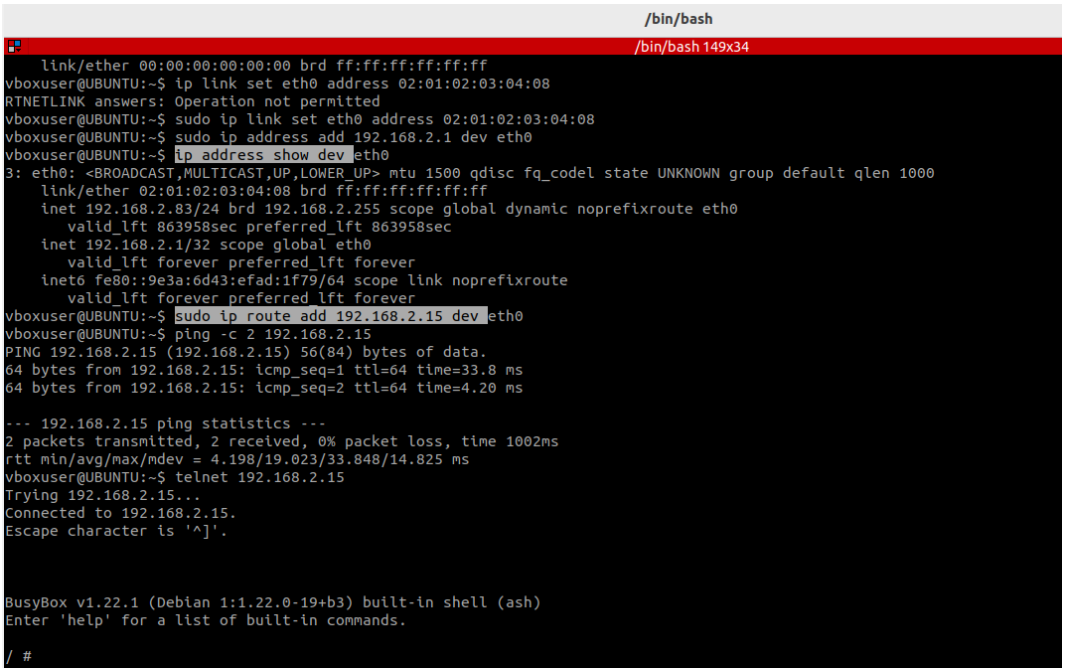

Figura 3.5: Comunicazione con il dispositivo tramite connessione telnet

Durante il processo di build di “halium-boot.img” la configurazione di CONFIG_VT viene sovrascritta da quanto specificato dal file Kconfig nella directory /drivers/tty, il quale disabilita automaticamente il parametro impostato nel file di configurazione principale.
Dopo aver cambiato l’impostazione predefinita, ho ricostruito e installato la nuova immagine “halium-boot.img”.

Il processo di porting verso Ubuntu Touch fallisce in questo momento, infatti, nonostante il sistema operativo venga caricato in modo corretto all’avvio, il dispositivo non riesce a completare il processo di boot entrando in uno stato di bootloop, cioè innescando un riavvio automatico continuo, a causa di un kernel panic del quale non è stato possibile reperire un log sufficientemente dettagliato.

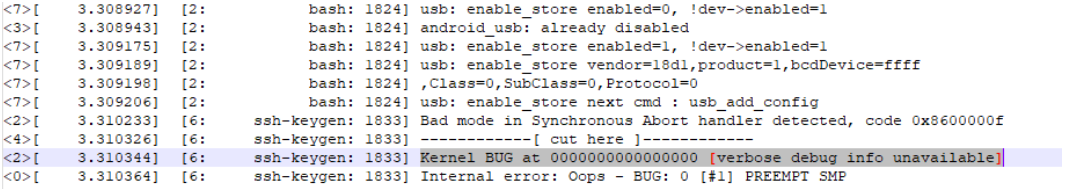

Figura 3.6: Messaggio di Kernel Panic nel file log “last_kmsg”

3.4 Porting di PostmarketOS

Lo sviluppo del porting del sistema operativo Open Source PostmarketOS inizia con l'installazione dell'applicazione Pmbootstrap sul computer host. Pmbootstrap è un software a riga di comando, installabile su qualsiasi distribuzione Linux, che permette di costruire i vari pacchetti del sistema operativo, di creare le immagini di sistema e di eseguire il flash sul dispositivo mobile.
Prima di iniziare, Pmbootstrap deve essere inizializzato con il comando "pmbootstrap init". Durante questa operazione, si specificano varie opzioni, tra cui il produttore dello smartphone, il suo nome, l'interfaccia da utilizzare e la cartella dove verranno salvati i file prodotti. Inoltre, verrà richiesto anche il metodo di flash utilizzato dal cellulare, nel mio caso specifico ho selezionato "heimdall" piuttosto che "fastboot" poichè quest'ultima modalità non è utilizzata dai dispositivi Samsung.
Per quanto riguarda l'interfaccia utilizzata, per il dispositivo a5xelte ho scelto Xfce4 che, anche se poco ottimizzata per schermi di dimensioni ridotte, viene consigliata per i device che utilizzano un kernel non "mainline".

3.4.1 Compilazione Kernel

Per compilare il kernel bisogna prima apportare alcune modifiche ai due file APKBUILD presenti nelle directory di Pmbootstrap clonato da git:
/pmaports/device/testing/device-samsung-a5xelte e
/pmaports/device/testing/linux-samsung-a5xelte.
All'interno di entrambi i file APKBUILD bisogna cambiare alcuni parametri in base all'architettura del dispositivo che si sta utilizzando e, per quello contenuto in linux-samsung-a5xelte, aggiungere la sorgente GitHub del kernel dello smartphone nella sezione #Source.
Per ottenere, invece, la configurazione del kernel, si può utilizzare il defconfig presente nel repository del kernel (come nel caso di Ubuntu Touch). Il file dovrà essere salvato come "config-samsung-a5xelte.aarch64" nella cartella linux-samsung-a5xelte. Nel mio caso specifico, il device a5xelte è a 64bit, perciò, si utilizza il formato aarch64, se fosse stato a 32bit avremmo usato il formato armhf.

Dopo aver apportato le dovute modifiche, ho utilizzato il comando “pmbootstrap checksum linux-samsung-a5xelte” per effettuare, prima, il download, poi, il controllo dell’integrità dei dati di quanto riportato nella sezione #Source di APKBUILD.

Ora che tutto è disponibile localmente, si può procedere con la personalizzazione del kernel che, come per Ubuntu Touch, deve essere configurato in modo specifico per l’utilizzo con PostmarketOS, con alcuni parametri diversi rispetto a un kernel per il sistema operativo Android.
Per determinare le impostazioni necessarie si utilizza il comando “pmbootstrap kconfig check” che fornirà un elenco delle opzioni da abilitare o disabilitare.
Con il comando “pmbootstrap kconfig edit” si apre l’editor di configurazione del kernel: attraverso un menù diviso per categorie, è possibile modificare le varie opzioni in base all’output del comando precedente o alla wiki ufficiale.

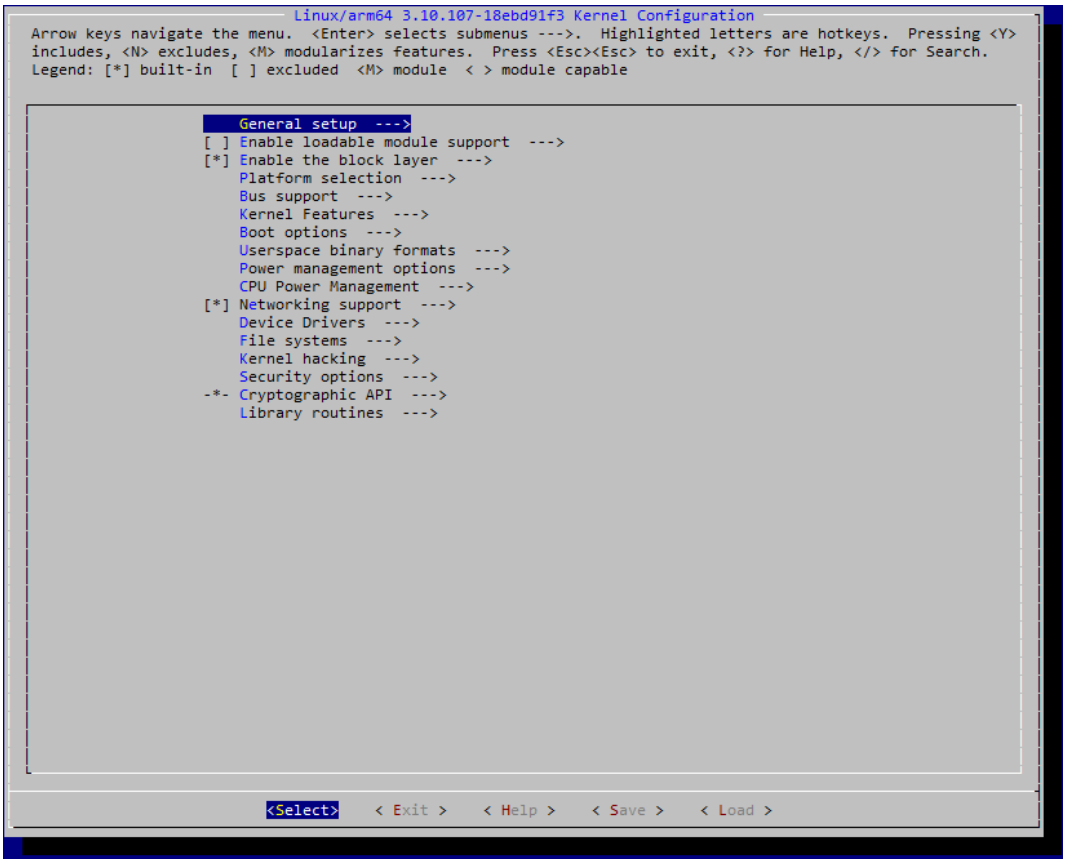

Figura 3.7: Menù di configurazione del kernel: ogni opzione è contenuta nell’apposita categoria

Dopo aver verificato che tutti i parametri siano impostati correttamente, anche rieseguendo il comando di controllo, si può proseguire con la compilazione del kernel con il comando "pmbootstrap build linux-samsung-a5xelte". Durante il processo, potrebbero presentarsi alcuni errori, che verranno opportunamente segnalati dal programma nel suo log.
Per risolvere questi problemi, nel mio caso specifico, ho importato delle patch sviluppate dagli utenti direttamente nella cartella del file APKBUILD. In quest'ultimo file è importante aggiungere il nome di ogni patch nella categoria #Source.
Poichè sono stati sviluppati molti porting di PostmarketOS per dispositivi con architetture diverse, gli errori riscontrabili possono essere risolti cercando ciò che manca all'interno delle cartelle degli altri device per poi provare a ricompilare finché il procedimento non va a buon fine.

3.4.2 Configurazione file "deviceinfo"

Adesso che il kernel è stato costruito correttamente, mi sono spostato all'interno della cartella device-samsung-a5xelte. Al suo interno si trova il file "deviceinfo" che contiene una serie di variabili da configurare correttamente. Queste riguardano, come dice anche il nome del file stesso, le informazioni del dispositivo: il nome, il produttore, il tipo di architettura, informazioni sull'hardware e sul bootloader. Queste ultime, in particolare, sono state ricavate analizzando l'immagine di sistema boot.img di LineageOS costruito per il modello a5xelte con il comando "pmbootstrap bootimg_analyze".
Nello specifico, sono stati aggiustati:

- l'altezza e la larghezza dello schermo impostando deviceinfo_screen_width="1080" e deviceinfo_screen_height= "1920";
- deviceinfo_flash_pagesize="2048" che specifica la dimensione della pagina di memoria flash;
- deviceinfo_flash_offset_base="0x10000000";
- deviceinfo_flash_offset_kernel="0x00008000";
- deviceinfo_flash_offset_ramdisk="0x01000000";

- deviceinfo_flash_offset_second="0x00f00000";
- deviceinfo_flash_offset_tags="0x00000100";

le ultime cinque variabili sono necessarie per indicare dove caricare, nella memoria flash, ciascuna parte del firmware.

Ora si può creare il pacchetto specifico per il mio dispositivo: ho eseguito "pmbootstrap checksum device-samsung-a5xelte" per verificare l'integrità dei file e "pmbootstrap build device-samsung-a5xelte" per la compilazione.

3.4.3 Installazione PostmarketOS

Per installare PostmarketOS sul dispositivo si possono seguire, come per Ubuntu Touch, varie strade. In questo caso non ho utilizzato né il flash con fastboot, non fattibile, né il flash con Odin/Heimdall per una limitazione della macchina virtuale, ma ho sfruttato l'installazione tramite ADB sideload con la recovery TWRP.
Il sideload è una funzionalità di Android Debug Bridge (ADB) che consente di scaricare il file zip contenente il sistema operativo tramite la connessione USB dal computer al dispositivo e di installarlo.
Il primo step è creare l'immagine in formato zip e di esportarla grazie a:

- "pmbootstrap install –android-recovery-zip";
- "pmbootstrap export".

Spostandosi sullo smartphone, si avvia la modalità recovery dove si smontano tutte le partizioni e si avvia il sideloading tramite l'opzione "ADB Sideload".
Dopo aver fatto ciò, si collega il telefono al PC tramite USB, si apre il terminale sul computer host nella directory contenente il file così creato. Da qui, si esegue "adb sideload pmos-samsung-a5xelte.zip".
Se non compaiono messaggi di errore, PostmarketOS è così installato sul dispositivo.

Figura 3.8: Installazione tramite “ADB Sideload” di PostmarketOS sul dispositivo Samsung a5xelte

3.5 Avvio di PostmarketOS

Dopo aver completato l'installazione di PostmarketOS sul dispositivo a5xelte, è incoraggiante vedere il logo verde del software durante l'avvio.
La presenza del logo di PostmarketOS, infatti, costituisce un importante passo in avanti verso l'obiettivo di realizzare un porting funzionante perché significa che l'immagine del sistema operativo è presente nello smartphone e che il kernel è stato avviato con successo.

Figura 3.9: Avvio di PostmarketOS: il dispositivo, non ancora configurato, rimane fermo sul logo

Tuttavia, allo stato attuale il device non è ancora utilizzabile perché occorrono ulteriori configurazioni.
Il passo successivo consiste nell'effettuare la configurazione del dispositivo a5xelte tramite una connessione USB, utilizzando il protocollo Secure Shell (ssh); è importante notare che questi passaggi possono variare per ogni dispositivo e potrebbero

essere necessari interventi diversi.
Per verificare se lo smartphone a5xelte è correttamente connesso al PC ho eseguito il comando "ip a" per visualizzare l'elenco dei dispositivi collegati. L'a5xelte è stato riconosciuto correttamente e sono specificati i suoi indirizzi MAC e IP dinamico, così come la maschera di rete, utili per la comunicazione tramite USB.
Ora, con il comando "ssh user@172.16.42.1", mentre il cellulare è "bloccato" sul logo, è possibile instaurare una connessione: dopo aver inserito la password impostata durante il processo di creazione dell'immagine di sistema, apparirà il messaggio "Welcome to postmarketOS!" e diventa possibile eseguire comandi all'interno della shell del dispositivo.

Ogni volta che lo smartphone viene acceso, rimane fermo alla schermata del caricamento con il logo di PostmarketOS.
Dal computer ho quindi eseguito nella shell del device il comando "sudo startx", in questo modo ho avviato l'interfaccia grafica e ho avuto accesso all'ambiente Linux Alpine come utente superuser. Tramite questa istruzione, però l'interfaccia che si presenta è quella tipica della versione desktop del sistema operativo, quindi con cursore del mouse visibile, che può essere spostato tramite il touchscreen, ed elementi grafici come icone e testo di dimensioni molto ridotte e di difficile interazione.
L'obiettivo principale di questa operazione era testare il funzionamento del display e del touchscreen e ho potuto constatare che erano stati correttamente implementati.

Figura 3.10: Schermata di Home di PostmarketOS avviato usando il comando “sudo startx”

È stato anche possibile stabilire una connessione ad Internet tramite il cavo USB, il quale mi ha permesso di scaricare alcuni importanti pacchetti che verranno dettagliati meglio in seguito.
Sulla mia macchina host con Ubuntu ho dovuto eseguire i seguenti comandi per permettergli di funzionare come un router:

- “sysctl net.ipv4.ip_forward=1” abilita l’IP forwarding;
- “iptables -A FORWARD -m conntrack –ctstate ESTABLISHED, RELATED -j ACCEPT” consente di accettare i pacchetti in transito che sono già stati stabilitati o sono correlati a una connessione già stabilita;
- “iptables -A FORWARD -s 172.16.42.0/24 -j ACCCEPT” consente di accettare tutti i pacchetti provenienti dalla subnet specificata;

- "iptables -A POSTROUTING -t nat -j MASQUERADE -s 172.16.42.0/24" modifica gli indirizzi sorgente dei pacchetti provenienti dalla subnet specificata in modo che sembrino provenire dalla macchina host, consentendo loro di passare per accedere alla rete esterna;
- "iptables -save" salva quanto configurato in precedenza per essere utilizzato anche dopo un riavvio del sistema.

Sul dispositivo a5xelte, invece, è necessario eseguire i seguenti comandi nel terminale per configurare il NAT sulla connessione USB tethering tra il dispositivo e la macchina host, in modo che i pacchetti possano essere instradati dall'interfaccia USB alla rete esterna:

- "nft add table inet nat" crea una nuova tabella NAT nel namespace "inet";
- "nft 'add chain inet nat postrouting type nat hook postrouting priority 100;'" fa in modo che le regole vengano applicate dopo che i pacchetti sono stati inviati da un'interfaccia;
- "nft add rule inet nat postrouting iifname "usb*" masquerade" aggiunge una regola alla catena "postrouting" per effettuare la mascheratura degli indirizzi IP sorgente dei pacchetti provenienti dall'interfaccia USB, in modo che sembrino provenire dall'host e possano essere instradati sulla rete esterna;
- "nft add rule inet filter forward ct state established, related accept" aggiunge una regola alla catena "forward" della tabella "filter" per accettare tutti i pacchetti che appartengono agli stati associati a connessioni già stabilite;
- "nft add rule inet filter forward iifname "usb*" accept" aggiunge una regola alla catena "forward" della tabella "filter" per accettare tutti i pacchetti provenienti dall'interfaccia USB.

Dopo aver configurato queste regole, quando ho avuto bisogno di Internet ho eseguito "ip route add default via 172.16.42.2 dev usb0" per inoltrare i pacchetti attraverso l'interfaccia usb0.
Ho verificato, infine, la connessione ad Internet sul cellulare effettuando il ping verso Google.it e scaricando il pacchetto hello-world.

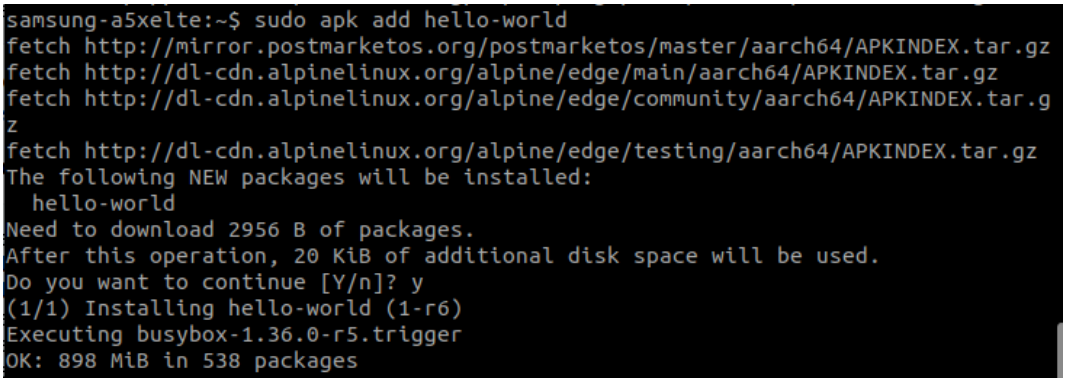

Figura 3.11: Download del pacchetto hello-world sul dispositivo connesso ad Internet tramite USB

Per installare i diversi pacchetti verrà utilizzato il comando “apk add” inserendo, alla fine, il nome del pacchetto.
La tastiera virtuale non è presente, perciò, è impossibile scrivere sul terminale o in qualsiasi altra finestra del sistema operativo, per questo motivo ho scaricato il pacchetto “onboard” che riconosce quando l’utente prova a inserire del testo e apre automaticamente una tastiera a schermo.
Implementando questa funzione si può continuare il setup del dispositivo.

Come già accennato precedentemente, per adesso si può avviare l’interfaccia grafica di PostmarketOS solo attraverso “sudo startx”, il che rende impossibile utilizzare lo smartphone senza un computer collegato. Per modificare questo comportamento, occorre configurare un display manager, un programma che viene utilizzato per gestire l’accesso degli utenti al sistema fornendo una schermata di login dove inserire le proprie credenziali e che viene caricato all’accensione del device.
Il display manager che ho scelto è LightDM in quanto molto leggero e poco impattante sulle prestazioni del telefono.
Per installare in modo completo LightDM occorre scaricare due pacchetti con il comando “apk add lightdm lightdm-gtk-greeter”: LightDM Greeter è essenziale per mostrare all’utente l’interfaccia grafica sulla quale inserire i propri dati e, oltre a questo, offre alcune possibilità di personalizzazione per cambiare sfondo e tema.
Poichè il software non viene configurato automaticamente, ho dovuto modificare il file denominato “lightdm.conf” all’interno della directory /etc/lightdm. Questo

file descrive in maniera dettagliata, suddividendole in sezioni, tutte le opzioni di configurazione.
Iniziando dalla sezione della configurazione generale individuata dalla voce [LightDM], ho inserito il parametro "logind-check-graphical=false" in modo che il display manager venga caricato al termine del boot di PostmarketOS.
Sono passato, poi, ad aggiungere altri parametri all'interno della sezione che controlla tutte le Seat, indicata come [Seat:*]. Una Seat è costituita da un display e da un set di input associati, nel mio caso la Seat principale è il display ed il touchscreen ma potrebbe essercene un'altra nel caso si utilizzasse un monitor esterno, un mouse o una tastiera.
Ho aggiunto le seguenti righe: "greeter-session = lightdm-gtk-greeter" per specificare il pacchetto Greeter da utilizzare per l'interfaccia utente della schermata di login, "greeter-hide-users = false" per visualizzare tutti gli utenti che possono autenticarsi al sistema.
In questa sezione è possibile anche abilitare l'autologin oppure la sessione per gli utenti ospiti.
Specificatamente al Greeter, occorre modificare il file "lightdm-gtk-greeter.conf" che contiene i parametri che regolano l'aspetto grafico, i fonts e la grandezza del testo, la posizione della finestra di login, la barra degli strumenti e la sicurezza.
Quello che ho aggiunto riguarda l'utilizzo della tastiera virtuale che al momento del login ancora non viene caricata dal sistema operativo rendendo impossibile l'accesso: il parametro "keyboard = onboard" specifica quale applicazione avviare per mostrare la tastiera, "keyboard-position="50%, center -0;100% 25%"" specifica la posizione e "ally-states=+keyboard" serve per renderla disponibile all'avvio.
Con queste ultime aggiunte è conclusa la configurazione di LightDM e all'avvio del cellulare non si rimarrà più bloccati al logo di PostmarketOS ma verrà mostrata la finestra di login.

Figura 3.12: Schermata di Login di LightDM con tastiera virtuale “onboard”

Allo stato attuale, con il porting di PostmarketOS per a5xelte è possibile utilizzare un sistema Alpine Linux in modo portatile pur mancando di funzioni fondamentali. I pacchetti sviluppati per utilizzare il modem includono “Calls” per effettuare chiamate, anche VoIP, “Chatty” per inviare messaggi e “Gnome-contacts” per la gestione della rubrica. Purtroppo, non è possibile sfruttare questo tipo di software in quanto le funzionalità del modem non sono disponibili a causa della mancanza di un firmware adatto.

Inoltre, anche il Wi-Fi è tra le funzioni mancanti; tuttavia, è possibile navigare nel web tramite connessione USB con il browser Firefox.

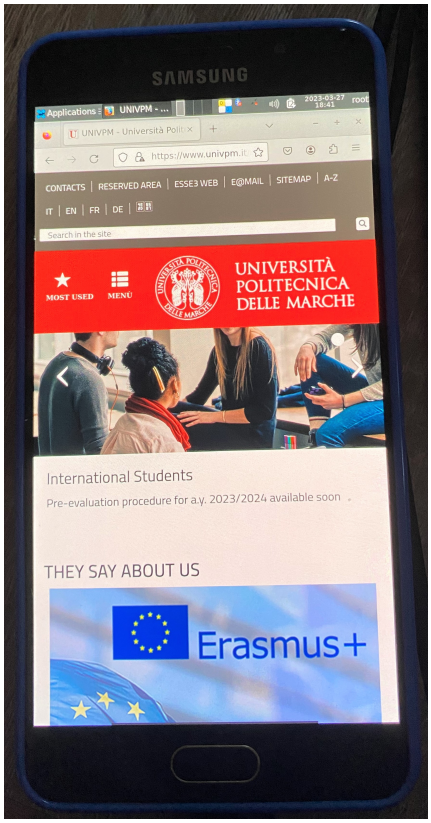

Figura 3.13: Utilizzo del Browser Web Firefox in PostmarketOS

Il Porting di PostmarketOS per il dispositivo a5xelte, essendo datato e utilizzando un SoC Samsung Exynos 7580 per il quale non si dispongono di informazioni utili allo sviluppo, non implementa alcune funzioni rilevanti per permettere al telefono di svolgere appieno il suo scopo. Oltre a quanto specificato in precedenza, mancano:

- Audio, non funzionante, nonostante sia possibile modificare il volume tramite i tasti hardware;
- Bluetooth, per il quale il sistema non riesce a rilevare il modulo dedicato;
- fotocamera e sensori, per i quali non è possibile installare firmware proprietari per questa tipologia di componente.

Capitolo 4

Conclusioni e Sviluppi Futuri

4.1 Conclusioni

La tesi sul porting di sistemi operativi Open Source su un dispositivo mobile ha raggiunto risultati misti. Mentre il porting di Ubuntu Touch non è andato a buon fine a causa di un errore di cui non si dispongono di log esaustivi e di mancanza di supporto della comunità, il porting di PostmarketOS è stato installato con successo, seppur con alcune funzionalità mancanti.
Il progetto ha dimostrato che l'adattamento di un sistema operativo per un device specifico è un processo molto complesso che coinvolge diversi fattori.
La scelta del dispositivo ha sicuramente influito sul risultato raggiunto, in quanto, trattandosi di un device Samsung, il suo hardware è gestito da software proprietario del quale non è possibile reperire il codice sorgente.
Inoltre, il supporto della comunità per dispositivi datati come quello utilizzato è limitato e, nel mio caso, chi aveva tentato il porting era rimasto fermo ai primi passi dello sviluppo. Gli sviluppatori sono principalmente concentrati, infatti, sugli smartphone di ultima generazione per migliorare il sistema operativo in base alle nuove tecnologie e, soprattutto, su quelli che adottano un hardware "Open Source" per il quale lo sviluppo è agevolato.
In generale, si può concludere affermando che, nonostante il cellulare a5xelte non abbia potuto essere un esempio funzionale per dimostrare come un sistema operativo diverso da Android o iOS possa comunque essere usato nella realtà quotidiana, l'utilizzo di un software Open Source è una scelta da prendere in considerazione.
Durante lo sviluppo del progetto, infatti, ho avuto sempre disponibile e consultabile

il codice di ogni parte dei due sistemi operativi e se avessi avuto la necessità, lo avrei potuto modificare liberamente.
Ovviamente non tutti gli utenti hanno la capacità o il bisogno di effettuare tali modifiche ma, almeno, si ha la conoscenza di come i propri dati e la propria privacy vengono gestiti.

4.2 Sviluppi Futuri

Sotto l'aspetto degli sviluppi futuri, quello dei sistemi operativi Open Source è un mondo molto variegato e può adattarsi alle esigenze di ogni tipo di utente per questo le opportunità di business che può offrire sono molteplici.
Invece di mirare a sostituire Android e iOS, è possibile puntare a nicchie di utenti interessate a un sistema operativo personalizzato e specifico per le loro esigenze. Le aziende spesso gestiscono grandi quantità di dati sensibili e riservati dei propri clienti, dipendenti e partner commerciali così come di progetti riservati e non divulgabili che spesso a causa di problemi di sicurezza vengono pubblicati in modo non autorizzato. Per questo motivo, questa potrebbe essere un'area sulla quale concentrarsi con l'obiettivo di fornire un sistema trasparente e con funzionalità avanzate di sicurezza per i dispositivi utilizzati internamente, al fine di evitare fughe di dati e accessi non autorizzati.
Un sistema operativo Open Source può essere proposto anche a coloro che cercano la massima privacy e che non vogliono lasciare ad altri il controllo delle proprie informazioni. Tra questi utenti possono esserci per esempio, in paesi con governi autoritari, giornalisti che lavorano su temi sensibili oppure attivisti per i diritti umani e la libertà di parola ed espressione che possono essere soggetti a sorveglianza e monitoraggio da parte delle autorità.
Ciò che può affascinare l'utente comune che, spesso, poco si interessa dell'aspetto della sicurezza, può essere una durata maggiore della batteria: l'unione di un sistema operativo personalizzato e leggero con un hardware efficiente è la soluzione giusta per chi ha bisogno di utilizzare uno smartphone per lunghi periodi di tempo senza poterlo ricaricare.
PostmarketOS, essendo un sistema operativo mobile altamente personalizzabile e

sicuro, è la base perfetta per raggiungere questa vasta utenza. Potrebbe essere considerata anche la progettazione di un dispositivo su misura per avvicinarsi al pubblico di massa e mantenere un'interfaccia user-friendly che è anche il motivo per il quale Android e iOS hanno avuto tanto successo.

Infine, la creazione di un app store dedicato potrebbe essere una risorsa preziosa per gli utenti, offrendo loro un facile accesso a un'ampia gamma di applicazioni e funzionalità aggiuntive aumentando, così, l'attrattiva del device agli occhi del pubblico e degli sviluppatori, invogliati a rilasciare software compatibile anche con PostmarketOS.

In conclusione, si può dire che il presente progetto di porting di sistemi operativi mobile Open Source costituisce una base per comprendere i vantaggi e le problematiche dello sviluppo e dell'installazione di tali software con l'obiettivo, anche, di promuovere l'Open Source come una valida alternativa ai più famosi sistemi proprietari.

Sitografia

[1] Etica Open Source. https://virgo29.it/tutorial/tutorialvideo/open-source-perche-sceglierlo-etica/.

[2] Open Source. https://www.redhat.com/it/topics/open-source/what-is-open-source.

[3] Open Source Wikipedia. https://it.wikipedia.org/wiki/Open_source.

[4] StatCounter. https://gs.statcounter.com/os-market-share/mobile/worldwide.

[5] Android Origini. https://www.tuttoandroid.net/android/.

[6] Sicurezza Android e iOS. https://www.outofbit.it/sicurezza-privacy-meglio-android-ios/.

[7] Duopolio Android iOS. https://www.pandasecurity.com/it/mediacenter/tecnologia/duopolio-android-ios/.

[8] Nuove norme UE per terze parti. https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/digital-markets-act-ensuring-fair-and-open-digital-markets_en.

[9] Alternative ad Android. https://www.lealternative.net/2020/03/25/alternative-ad-android/.

[10] Linux su smartphone PostmarketOS. https://www.wired.it/mobile/smartphone/2019/01/25/postmarketos-linux-smartphone/.

[11] UBPorts Foundation sito ufficiale. https://ubports.com/.

[12] UBPorts Documentazione. https://docs.ubports.com/en/latest/porting/.

[13] PostmarketOS Documentazione. https://wiki.postmarketos.org/wiki/Main_Page.

[14] Purism sito ufficiale. https://puri.sm/.

[15] Pine64 Wiki. https://wiki.pine64.org/wiki/Main_Page.

[16] Fairphone sito ufficiale. https://www.fairphone.com/it/.

[17] LightDM Documentazione. https://wiki.archlinux.org/title/LightDM.

[18] Ubuntu Touch Wikipedia. https://en.wikipedia.org/wiki/Ubuntu_Touch.

[19] Halium Documentazione. https://halium.org/.