Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Zajc

Svoboda na dotik: Ali je popolnoma odprtokoden telefon možen?

osnutek diplomske naloge

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. David Jelenc

Ljubljana, 2026

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Jan Zajc
Naslov: Svoboda na dotik: Ali je popolnoma odprtokoden telefon možen?
Vrsta naloge: Diplomska naloga na visokošolskem strokovnem programu prve stopnje Računalništvo in informatika
Mentor: viš. pred. dr. David Jelenc

Opis:
Opis diplome

Title: Freedom at Your Fingertips: Is a Fully Open-Source Phone Possible?

Description:
Opis diplome v angleščini

Kazalo

Povzetek

Naslov: Svoboda na dotik: Ali je popolnoma odprtokoden telefon možen?

Avtor: Jan Zajc

placeholder

Ključne besede: odprtokodni telefon, odprtokodna programska oprema, pametni telefoni, mobilni operacijski sistemi, zasebnost.

Abstract

Title: Freedom at Your Fingertips: Is a Fully Open-Source Phone Possible?

Author: Jan Zajc

placeholder

Keywords: open-source phone, open-source software, smartphones, mobile operating systems, privacy.

Poglavje 1

Uvod

placeholder

Poglavje 2

Pregled področja: mobilni operacijski sistemi in odprtokodne alternative

Mobilni telefoni so se iz komunikacijskih naprav razvili v splošnonamenska računalniška orodja, ki jih uporabljamo za delo, zabavo, navigacijo, avtentikacijo in komunikacijo. V praksi to pomeni, da je operacijski sistem pametnega telefona postal ena izmed ključnih komponent digitalnega življenja uporabnika. Trenutno stanje na trgu je močno centralizirano. Mobilni ekosistem je v veliki meri razdeljen med Android in iOS, kar ima neposredne posledice za svobodo uporabnika, zasebnost in življenjsko dobo naprav. V tem poglavju najprej opišemo ključne lastnosti obeh prevladujočih platform, nato pa pregledamo glavne odprtokodne pristope in njihove omejitve.

2.1 Terminologija

V poglavju uporabljamo nekaj izrazov, ki se pri mobilnih sistemih pogosto pojavljajo v angleščini ali pa imajo v slovenščini več možnih prevodov. Da ne bi pri vsaki sekciji znova razlagali istih pojmov, spodaj na kratko zberemo ključne izraze in kratice. V nadaljevanju jih uporabljamo dosledno v istem

pomenu.

Android, iOS Dve prevladujoči mobilni platformi. Android je ekosistem naprav različnih proizvajalcev, iOS pa Applov operacijski sistem za iPhone.

Applova trgovina z aplikacijami (angl. *App Store*) Uradni kanal za distribucijo aplikacij na iOS.

Androidov odprtokodni projekt (AOSP) Odprtokodna osnova Androida (angl. *Android Open Source Project*), na kateri temeljijo številne distribucije in prilagoditve.

Izdelovalec originalne opreme (OEM) Proizvajalec končne naprave (angl. *Original Equipment Manufacturer*), ki tipično prilagodi sistem in programsko opremo za svojo strojno opremo.

Sistem na čipu (SoC; angl. *System-on-a-Chip*) Glavni čip v telefonu, ki običajno vključuje CPU, GPU, komunikacijske podsisteme in druge krmilnike (npr. Qualcomm Snapdragon).

Gonilnik Programska komponenta, ki operacijskemu sistemu omogoča uporabo strojne opreme (angl. *driver*).

Strojna programska oprema (angl. *firmware*) Programi, ki tečejo neposredno na strojnih komponentah (npr. kamera, Wi-Fi/Bluetooth, modem) in so pogosto lastniški.

Sklad dobavitelja (angl. *vendor stack*) Strojno-specifičen del programskega sklada, ki ga zagotovi dobavitelj čipovja ali proizvajalec naprave (gonilniki, strojna programska oprema, vmesniki).

Googlove mobilne storitve (GMS) Nabor lastniških Googlovih komponent na Androidu (angl. *Google Mobile Services*) (npr. trgovina z aplikacijami, storitve za obvestila, API-ji). Osrednji del predstavljajo sistemske storitve *Google Play Services* (pogosto poimenovane tudi *GMS Core*).

Programski vmesnik (API) Vmesnik, prek katerega aplikacije ali sistemske komponente uporabljajo funkcionalnosti drugih komponent (angl. *Application Programming Interface*).

Potisna obvestila (angl. *push notifications*) Mehanizem, kjer strežnik sproži obvestilo aplikaciji na napravi (pogosto prek posrednih storitev).

Preverjanje integritete (*SafetyNet*/*Play Integrity*) Mehanizmi, s katerimi ponudnik platforme preverja lastnosti naprave in okolja izvajanja (npr. certificiranost, nepooblaščene spremembe). SafetyNet je starejši pristop, Play Integrity pa novejši.

Upravljanje digitalnih pravic (DRM) Tehnike za nadzor dostopa do zaščitenih vsebin (angl. *Digital Rights Management*) (npr. video). V kontekstu Androida je pogost primer *Widevine*, ki uporablja različne varnostne ravni (npr. L1, L3).

Jedro (angl. *kernel*) Osrednji del operacijskega sistema. V mobilnem svetu je pogosto pomembno, ali naprava uporablja jedro iz glavne razvojne veje.

Linux V nalogi izraz Linux uporabljamo primarno za jedro operacijskega sistema. Kadar govorimo o celotnem sistemu (jedro + uporabniški prostor), uporabimo izraz "distribucija Linuxa" oziroma "sistem na osnovi Linuxa".

Uporabniški prostor (angl. *userspace*) Del sistema, kjer teče večina programov in storitev nad jedrom (npr. lupina, paketni upravljalnik, grafični vmesnik).

Jedro iz glavne razvojne veje (angl. *mainline kernel*) Jedro Linuxa, ki je del uradne (glavne) razvojne veje, ne pa posebej prilagojena različica proizvajalca.

Sistemska slika (ROM) V kontekstu Androida izraz ROM uporabljamo za sistemsko sliko oziroma distribucijo operacijskega sistema. Ločimo

npr. tovarniško sistemsko sliko (*stock ROM*) in prilagojeno sistemsko sliko (*custom ROM*). Kratica ROM izhaja iz angleškega izraza *read-only memory*, vendar v tem kontekstu ne označuje pomnilniškega čipa tipa ROM.

Skrbniški dostop (angl. *root*) Možnost izvajanja ukazov z najvišjimi privilegiji v sistemu.

Obnovitveno okolje (angl. *recovery*) Posebno zagonsko okolje za vzdrževanje naprave (namestitve, posodobitve, brisanje podatkov ipd.).

Prenos (angl. *porting*) Postopek prilagoditve operacijskega sistema ali distribucije na določeno strojno platformo oziroma napravo.

Modemski podsistem (angl. *baseband*) Komunikacijski podsistem za mobilna omrežja, pogosto ločen od glavnega procesorja in z lastno strojno programsko opremo.

Konvergenca (angl. *convergence*) Koncept, kjer ista naprava (telefon) v določenih načinih uporabe deluje tudi kot namizni računalnik.

Vmesni sloji (*libhybris*, *Halium*) Pristopi, ki omogočajo uporabo Androidovih strojno-specifičnih komponent (npr. knjižnic/gonilnikov) v okolju distribucij Linuxa na telefonih.

2.2 Duopol Android/iOS in posledice za uporabnike

Android in iOS predstavljata praktično celoten trg mobilnih operacijskih sistemov. Po podatkih StatCounter ima Android globalno večinski delež, iOS pa večino preostalega [36]. Podobno sliko kažejo tudi tržne analize prodajnih deležev [37]. Ne glede na natančne odstotke je ključna ugotovitev, da sta v praksi relevantni skoraj izključno ti dve platformi.

Takšna koncentracija moči pomeni, da sta Google in Apple posrednika med uporabnikom in napravo. Določata, kako se namešča programska oprema, kakšna so pravila za aplikacije, kakšni so varnostni mehanizmi in katere funkcionalnosti so sploh dostopne. Čeprav sta varnost in zanesljivost pomembni, tak model pogosto vodi v položaj, kjer se uporabnikove pravice (npr. nadzor nad sistemom, možnost prilagoditev, izbira storitev) podredijo poslovnim interesom ekosistema.

2.3 iOS: zaprt ekosistem in nadzor nad distribucijo aplikacij

iOS je zasnovan kot strogo nadzorovan sistem. Uporabnik praviloma namešča aplikacije prek uradne Applove trgovine z aplikacijami (*App Store*), sistem pa je zgrajen okoli modela podpisovanja kode in preverjanja integritete [12, 13]. Tak pristop ima prednosti, kot so enotna uporabniška izkušnja, relativno predvidljiv varnostni model in dolga programska podpora za naprave. Hkrati pa pomeni tudi, da ima uporabnik omejen vpliv na to, kaj se lahko poganja na napravi in kakšne posege v sistem je sploh možno izvesti.

Z vidika diplomske naloge je pomembno predvsem to, da iOS kot platforma ne omogoča resnične možnosti "popolnoma odprtokodnega telefona". Uporabnik niti teoretično nima možnosti zamenjati ključnih komponent sistema z odprtokodnimi alternativami. iOS je zato dober kontrast, saj pokaže, kako je videti maksimalno centraliziran mobilni ekosistem, kjer je nadzor nad napravo primarno v rokah proizvajalca [13].

2.4 Android: odprtokodna osnova in resničnost zaprtih komponent

Android se pogosto predstavlja kot odprtokoden sistem, saj temelji na Androidovem odprtokodnem projektu (AOSP) [8]. To je pomembna razlika v

primerjavi z iOS, ker AOSP omogoča javno dostopno izvorno kodo za velik del platforme in teoretično dopušča razvoj alternativnih distribucij.

V praksi pa tipična naprava z Androidom ni enaka AOSP. Večina naprav vsebuje:

- prilagoditve izdelovalca originalne opreme (OEM), ki pogosto vključujejo dodatne sistemske aplikacije;
- zaprtokodne gonilnike in strojno programsko opremo za posamezne komponente, ki so del sklada dobavitelja [6, 9];
- Googlove mobilne storitve (GMS), ki niso del AOSP in so na voljo le prek licence; pomemben del teh storitev predstavlja *GMS Core* oziroma *Google Play Services* [7, 3].

Posledica je, da je Android kot ekosistem hibrid. Odprtokodna osnova se v praksi nadgradi z več lastniškimi plastmi. To se najbolj pozna pri strojni podpori. Gonilniki za grafični pospeševalnik, kamero, mobilni modem in druge ključne komponente so pogosto zaprti ter vezani na konkretno jedro in različico sistema [6, 9]. Zaradi tega posodobitve niso zgolj vprašanje nove različice Androida, temveč tudi vprašanje združljivosti celotnega sklada gonilnikov in strojne programske opreme. Del problema Android poskuša nasloviti z modularizacijo sistemskih komponent (angl. *Project Mainline*), kjer je del posodobitev mogoče dostaviti ločeno od celotnega sistema [10].

Ena izmed posledic takšne arhitekture je tudi fragmentacija. Različni proizvajalci vzdržujejo svoje različice sistema in jedra, kar poveča stroške vzdrževanja in lahko negativno vpliva na hitrost ter trajanje zagotavljanja posodobitev [1, 27]. To je eden izmed razlogov, zakaj naprave pogosto dobijo le omejeno število večjih posodobitev, čeprav je strojna oprema še povsem zmogljiva. Kot minimalen referenčni okvir lahko uporabimo tudi zahteve programa za priporočene poslovne naprave (angl. *Android Enterprise Recommended*), ki za vključene naprave predvideva podporo trenutni izdaji in vsaj eni večji nadgradnji operacijskega sistema [2].

2.5 Akterji mobilnega ekosistema

Za razumevanje omejitev odprtokodnih mobilnih sistemov je ključno upoštevati, da telefon ni samo operacijski sistem in aplikacije. Pri tipični napravi sodeluje več akterjev:

- **Proizvajalec operacijskega sistema oziroma ekosistema** (Apple ali Google, posredno tudi skupnost AOSP);
- **Izdelovalec originalne opreme (OEM)** (npr. OnePlus), ki prilagodi sistem;
- **Proizvajalec sistema na čipu (SoC)** (npr. Qualcomm), ki zagotovi gonilnike in strojno programsko opremo;
- **Operaterji** (v določenih državah tudi certificiranje in omejitve);
- **Razvijalci aplikacij**, ki se pogosto zanašajo na lastniške programske vmesnike (API) in storitve (npr. potisna obvestila, *SafetyNet/Play Integrity*) [5, 4].

Poseben primer je **modemski podsistem**. Ta je pogosto ločen procesor ali podsistem s svojo strojno programsko opremo, ki je praviloma lastniška [24, 17]. Zaradi varnosti in regulative je razumljivo, zakaj je ta del izoliran, vendar z vidika odprtokodnosti predstavlja trdno omejitev. Tudi če zamenjamo operacijski sistem in aplikacije, ostaja osnovnopasovni podsistem ključen zaprt del naprave.

2.6 Alternativni Android ROM-i

Alternativni ROM-i za Android so pogost način, kako predvsem tehnično bolj podkovani uporabniki poskušajo povečati nadzor nad napravo. Najbolj poznan primer je LineageOS, poleg tega pa obstajajo še projekti, ki se posebej osredotočajo na zasebnost ali zmanjšanje odvisnosti od Googlovih storitev (npr. /e/OS, GrapheneOS) [26, 18, 22].

V primerjavi s tovarniškimi ROM-i alternative pogosto ponujajo:

- manj prednaložene odvečne programske opreme (angl. *bloatware*);
- bolj transparentne nastavitve zasebnosti;
- več možnosti prilagajanja (npr. skrbniški dostop, drugačno obnovitveno okolje, ter večji nadzor nad sistemskimi aplikacijami);
- v nekaterih primerih daljšo podporo za starejše naprave (odvisno od skupnosti in vzdrževanja) [16].

Ključna omejitev alternativnih ROM-ov je, da so še vedno vezani na isti problem kot tovarniški Android. Odvisni so od zaprtokodnih gonilnikov in strojne programske opreme, ki jih zagotavlja proizvajalec strojne opreme. Poleg tega se v zadnjih letih povečuje odvisnost aplikacij od lastniških storitev in mehanizmov preverjanja integritete (npr. *Play Integrity*), kar lahko na sistemih brez Googlovih storitev povzroča težave pri aplikacijah z višjimi varnostnimi zahtevami (npr. plačila, bančne aplikacije) [4, 21]. Pri zaščiti vsebin se uporabljajo mehanizmi upravljanja digitalnih pravic (DRM; angl. *Digital Rights Management*), kot je Widevine, kjer lahko razlike v varnostnih ravneh vplivajo na razpoložljive funkcionalnosti oziroma kakovost predvajanja [20, 42, 14, 15].

2.7 Mobilni Linux: postmarketOS, Ubuntu Touch in sorodni projekti

Mobilni Linux poskuša na telefon prinesti bolj klasičen model računalniškega sistema. V ospredju so standarden uporabniški prostor Linuxa, paketni upravljalnik, terminal in večja svoboda prilagajanja. Med bolj aktivnimi projekti sta postmarketOS in Ubuntu Touch.

Projekt postmarketOS gradi na distribuciji Alpine Linux in ima dolgoročno usmeritev v vzdržljivost ter ponovno uporabnost naprav [31]. S

tem poskuša zmanjšati odvisnost od specifičnih različic Androida in cilja na to, da bi naprave lahko uporabljale jedro iz glavne razvojne veje, kjer je to izvedljivo.

Ubuntu Touch razvija skupnost UBports in poskuša ponuditi uporabniku prijazen mobilni sistem z lastnim grafičnim vmesnikom. Pomemben del zasnove je koncept konvergence, kjer lahko ista naprava deluje kot telefon in kot namizni računalnik [41, 40]. Cilj je ponuditi alternativo Androidu, ki ni vezana na Googlov ekosistem in kjer je večina programske opreme odprtokodna.

Kljub obetom je resničnost mobilnega Linuxa še vedno močno odvisna od strojne podpore. Pri številnih napravah se za delovanje še vedno uporabljajo obhodi, kot so vmesni sloji za uporabo Android gonilnikov (npr. libhybris, Halium), kar pomeni kompromis med odprtostjo in praktično uporabnostjo [23, 25, 35]. To je tipičen primer konflikta. Ideal je "čist" Linux programski sklad, praksa pa pogosto zahteva uporabo obstoječih lastniških komponent.

2.8 Namenske Linux naprave

Poleg prenosov na obstoječe telefone Android obstajajo tudi namenske naprave, ki ciljajo na uporabnike, ki želijo odprtokodne alternative (npr. PinePhone, Librem 5) [29, 30, 32, 33]. Te naprave pogosto poskušajo izboljšati stanje tako, da izberejo strojno opremo z boljšo podporo v jedru Linuxa ali z bolj odprto dokumentacijo.

Prednost namenskih Linux telefonov je, da je programski sklad pogosto bolj konsistenten (jedro + uporabniški prostor + uporabniški vmesnik) in da skupnost razvoj usmeri na manjše število naprav. Slabost pa je, da se takšne naprave pogosto ne morejo kosati z najboljšimi telefoni z Androidom ali iOS glede zmogljivosti, kakovosti kamere in porabe energije. Zato ostajajo predvsem v niši, kar vpliva tudi na velikost skupnosti in hitrost razvoja.

2.9 Povzetek

Pregled področja pokaže, da mobilni ekosistem trenutno obvladuje duopol Android/iOS, kjer je uporabniška svoboda v veliki meri omejena z zaprtimi ekosistemi, lastniškimi komponentami in poslovnimi modeli. Android sicer ponuja odprtokodno osnovo (AOSP), vendar tipične naprave vsebujejo veliko zaprtih slojev, kar oteži dolgotrajno vzdrževanje in razvoj alternativ. Alternativni ROM-i za Android delno izboljšajo nadzor nad sistemom, vendar ostajajo odvisni od istih zaprtih gonilnikov in strojne programske opreme. Projekti mobilnega Linuxa ponujajo bolj radikalno alternativo, vendar je njihova praktična uporabnost še vedno močno omejena s strojno podporo in ekosistemom aplikacij. To neposredno motivira nadaljnja poglavja, kjer bodo najprej opredeljeni kriteriji "popolnoma odprtokodnega telefona", nato pa bodo analizirane obstoječe rešitve in empirično preizkušene.

Poglavje 3

Teoretična izhodišča

V prejšnjem poglavju smo pokazali, da je tudi pri sistemih z odprtokodno osnovo velik del mobilnega ekosistema odvisen od zaprtih komponent. Da lahko kasneje ocenimo, v kolikšni meri je "popolnoma odprtokoden telefon" v praksi izvedljiv, moramo najprej razjasniti osnovne pojme, sloje sistema in tipične omejitve. V tem poglavju opredelimo razliko med odprtokodno in prosto programsko opremo, na kratko predstavimo arhitekturo mobilnih sistemov ter izpostavimo vlogo gonilnikov in strojne programske opreme. Nato povežemo odprtost z varnostjo in zasebnostjo, pri čemer obravnavamo tudi tveganja skritih funkcionalnosti oziroma stranskih vrat (v literaturi pogosto označenih kot angl. *backdoor*).

3.1 Odprtokodna in prosta programska oprema

Pojma odprtokodna programska oprema (angl. *open source software*) in prosta programska oprema (angl. *free software*) se pogosto uporabljata kot sopomenki, vendar izhajata iz različnih poudarkov. Open Source Initiative (OSI) odprtokodnost definira s kriteriji, ki med drugim zahtevajo dostop do izvorne kode, možnost sprememb in redistribucije ter nediskriminatorne licence [28]. Free Software Foundation (FSF) pa v ospredje postavlja uporabniške svoboščine: program je prost, če omogoča (1) poganjanje za polju-

ben namen, (2) preučevanje delovanja, (3) prilagajanje ter (4) deljenje kopij, tudi spremenjenih [19]. V nadaljevanju zato uporabljamo tudi širši izraz prosta in odprtokodna programska oprema (FOSS; angl. *Free and Open Source Software*), kadar želimo zajeti oba vidika.

Pri ocenjevanju "odprtosti telefona" je pomembno, da ločimo licenco od samega dejstva, da je koda javno dostopna. V praksi se pojavljajo tudi projekti, kjer je izvorna koda objavljena, vendar licenca ne dopušča redistribucije ali sprememb; za tak model se pogosto uporablja izraz dostopna izvorna koda (angl. *source-available*). Za kriterije v tej diplomski nalogi zato ne zadošča zgolj javna objava kode, ampak zahtevamo tudi pravice, ki jih določajo licence, ter možnost dejanske uporabe spremenjenih različic (npr. namestitev na napravo).

3.1.1 Licence in posledice za mobilne sisteme

Licence določajo, kaj lahko uporabnik in razvijalec s kodo počneta. V grobem jih lahko razdelimo na:

- permisivne licence (npr. MIT, BSD, Apache 2.0), ki omogočajo tudi vključitev kode v zaprt izdelek, in

- licence z zahtevami deljenja izboljšav (angl. *copyleft*, npr. GPL), ki ob distribuciji pod določenimi pogoji zahtevajo, da so tudi izpeljana dela na voljo pod enako (ali združljivo) licenco.

Za telefone je posebej pomembno jedro Linux, ki je pod licenco GPLv2. To proizvajalcem pri distribuciji naprav nalaga obveznosti glede objave izvorne kode sprememb jedra, vendar še ne zagotavlja, da je odprt celoten sistem. V praksi lahko ostanejo zaprti gonilniki, strojna programska oprema in velik del uporabniškega prostora (npr. sistemske aplikacije ali dobaviteljske knjižnice).

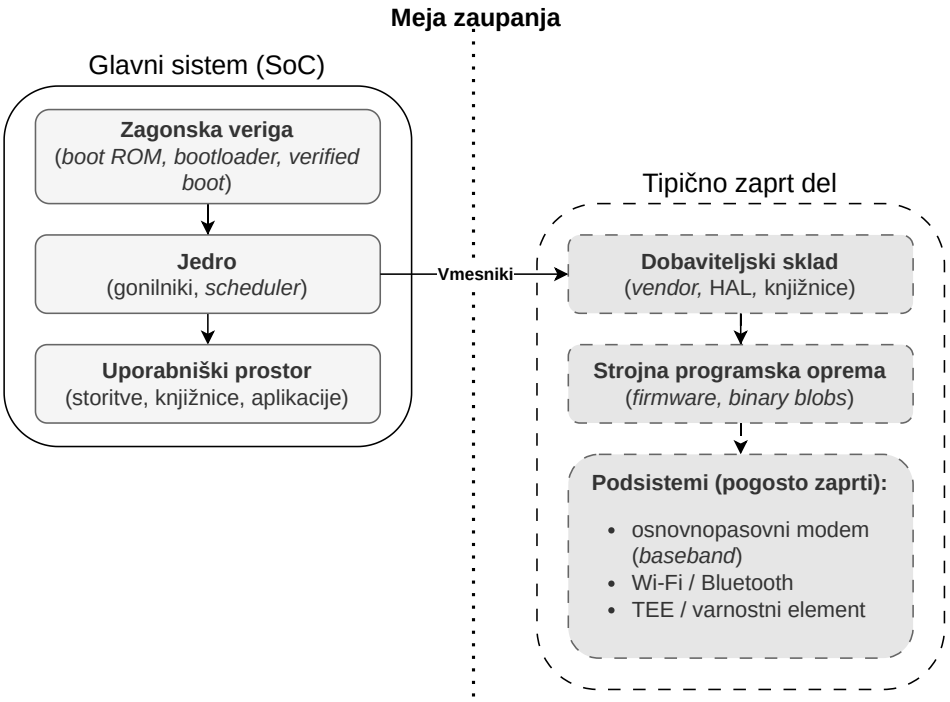
3.2 Arhitektura sistema: od zagona do uporabniškega prostora

Pametni telefon je sestavljen iz več podsistemov. Poleg glavnega procesorja (SoC) ima naprava pogosto ločene mikrokrmilnike ali koprocesorje za modem, Wi-Fi/Bluetooth, kamero in senzorje, pogosto pa tudi varnostne podsisteme, kot je zaupanja vredno izvajalno okolje (TEE; angl. *Trusted Execution Environment*). Ti deli praviloma uporabljajo lastno strojno programsko opremo in so pomemben del vprašanja zaupanja, ker so za uporabnika običajno netransparentni.

Na nivoju glavnega sistema ločimo:

- zagonsko verigo, ki določa, kaj se sploh sme zagnati (npr. zagonski nalagalnik) in preverjen zagon (angl. *verified boot*)),
- jedro z gonilniki in
- uporabniški prostor, kjer tečejo sistemske storitve, knjižnice in aplikacije.

Slika 3.1 povzema sloje sistema in tipične zaprte komponente, ki predstavljajo glavno omejitev pri popolnoma odprtokodnem telefonu.



Slika 3.1: Poenostavljena arhitektura telefona z vidika slojev sistema in tipičnih zaprtih komponent.

3.2.1 Linux: glavna razvojna veja jedra in vzdrževanje

Linux je družina sistemov, ki delijo isto jedro, nad njim pa lahko teče različna programska oprema uporabniškega prostora. Za dolgoročno vzdrževanje naprav je ključno, ali je podpora za strojno opremo vključena v glavno razvojno vejo jedra ali pa je vezana na proizvajalčevo prilagojeno različico jedra [38]. Če je podpora del glavne veje, lahko naprava praviloma dobiva varnostne popravke skupaj z ostalim ekosistemom. Če pa je odvisna od zaprtih ali slabo vzdrževanih sprememb dobavitelja, je življenjska doba programske podpore praviloma krajša.

3.2.2 Android: AOSP in ločitev dobaviteljskega sklada

Android temelji na jedru Linux, vendar ima specifičen uporabniški prostor in nabor sistemskih storitev, ki skupaj tvorijo platformo [8]. Za odprtost je pomembna meja med generičnim sistemom in strojno specifičnimi deli. Projekt Treble je uvedel bolj jasno ločitev na particije in vmesnike (npr. ločen `system` in `vendor`), s čimer sistem teoretično omogoča posodabljanje neodvisno od dobaviteljskega sklada [6, 9]. Kljub temu dobaviteljski del pogosto vsebuje zaprt nabor knjižnic, implementacije abstrakcijskega sloja strojne opreme (HAL), gonilnike in strojno programsko opremo, brez katerih naprava ne deluje.

3.3 Gonilniki, strojna programska oprema in binarni dodatki

Pri telefonih se pogosto srečamo s situacijo, kjer je del kode odprtokoden, ključne funkcionalnosti pa so dobavljene v obliki binarnih dodatkov. Pomembno je ločiti med:

Gonilniki kodo, ki teče v jedru ali uporabniškem prostoru in omogoča komunikacijo s strojno opremo.

Strojno programsko opremo kodo, ki teče na sami komponenti (npr. Wi-Fi čip, digitalni signalni procesor ali modem) in jo sistem zgolj naloži ali upravlja.

Tudi če so gonilniki odprtokodni, je strojna programska oprema lahko popolnoma zaprta, brez možnosti revizije ali popravkov. To je ena glavnih tehničnih ovir za telefon, kjer bi bil celoten programski sklad pregledljiv in zamenljiv. Najbolj problematični so podsistemi z visokimi privilegiji ali neposrednim vplivom na komunikacijo (npr. osnovnopasovni modem in brezžični vmesniki). Raziskave kažejo, da se ranljivosti in napadi pogosto selijo prav v zaprt dobaviteljski sklad in strojno programsko opremo, kjer je preglednost manjša in so orodja za analizo omejena [24, 17].

3.4 Odprtost, varnost in zasebnost

Odprta koda lahko prispeva k varnosti, ker omogoča neodvisen pregled in hitrejše odkrivanje napak. Vendar odprtost sama po sebi ni garancija; varnost je predvsem rezultat procesa (revizije, testiranja, odzivnost na ranljivosti) in nadzora nad tem, kaj se na napravi dejansko izvaja. Pri telefonih zato poleg licence ocenjujemo tudi dobavno verigo, mehanizme posodabljanja ter stopnjo zaprtih komponent.

3.4.1 Stranska vrata in dobavna veriga

Stranska vrata razumemo kot namerno dodano ali prikrito funkcionalnost, ki omogoča dostop mimo običajnih varnostnih mehanizmov. Pri zaprtih komponentah je problem očiten: uporabnik in neodvisni raziskovalci težko preverijo, ali takšna funkcionalnost obstaja. Tudi pri odprtokodni programski opremi pa ostaja pomemben napadni vektor dobavna veriga in orodja za izgradnjo. Klasičen primer je Thompsonov opis, kako lahko kompromitiran prevajalnik vstavi stranska vrata v program tudi takrat, ko je izvorna koda navidezno "čista" [39]. Za oceno zaupanja je zato smiselno gledati širše od same izvorne kode.

Pri telefonih se pomisleki o stranskih vratih najpogosteje vežejo na:

- zaprt dobaviteljski sklad na komunikacijskih čipih (modem, Wi-Fi/Bluetooth),
- privilegirane varnostne podsisteme (npr. TEE in varnostni element) in
- lastniške sistemske storitve s širokimi dovoljenji (npr. telemetrija).

3.4.2 Integriteta zagona in ponovljive gradnje

Moderni telefoni pogosto uporabljajo mehanizme preverjenega zagona, kjer je zagonska veriga kriptografsko vezana na podpisane sistemske slike. To izboljša zaščito pred zlonamernimi spremembami, hkrati pa lahko omeji lastnika naprave, če je zagonski nalagalnik zaklenjen in proizvajalec ne omogoča nalaganja alternativnih podpisov [11]. Za sisteme, ki ciljajo na odprtost, je zato pomembno ravnotežje: želimo visoko integriteto, vendar tudi možnost, da lastnik zakonito naloži in podpiše lastno zgradbo sistema.

Dodaten sloj zaupanja predstavljajo ponovljive gradnje (angl. *reproducible builds*), kjer lahko neodvisna stran iz iste izvorne kode izdela identične binarne pakete [34]. S tem zmanjšamo tveganje, da bi bila uradna izdaja kompromitirana na poti od kode do binarne slike. Pri telefonih je to posebej relevantno, ker uporabniki praviloma nameščajo vnaprej zgrajene sistemske slike, ne pa da bi sistem prevajali sami.

3.4.3 Zasebnost kot posledica arhitekture

Zasebnost pri telefonih ni samo vprašanje aplikacij, ampak tudi sistemske arhitekture. Če naprava temelji na zaprtih storitvah in naborih API-jev, uporabnik težko oceni, kateri podatki se pošiljajo in kdaj. Alternativni sistemi lahko zmanjšajo odvisnost od centraliziranih storitev, vendar se pri tem pogosto pojavi kompromis pri združljivosti aplikacij, zlasti pri aplikacijah z višjimi varnostnimi zahtevami, ki uporabljajo mehanizme preverjanja integritete [4].

3.5 Vzdrževanje in življenjski cikel naprav

Pri oceni odprtokodnega telefona je praktična uporabnost močno povezana z vzdrževanjem. Če naprava ne dobiva varnostnih popravkov, odprtost hitro postane zgolj formalna lastnost. Analize Android ekosistema kažejo velike razlike v dolžini podpore in zamudah pri varnostnih posodobitvah med proizvajalci [1]. V mobilnem Linuxu se ta problem pogosto rešuje z uporabo glavne razvojne veje jedra in standardnega uporabniškega prostora, vendar je cena praviloma slabša strojna podpora in manj zrel ekosistem aplikacij.

V naslednjih poglavjih bomo na podlagi teh izhodišč postavili merljive kriterije, kaj v našem kontekstu pomeni "popolnoma odprtokoden telefon", nato pa bomo z njimi analizirali izbrane obstoječe rešitve.

Poglavje 4

Model "popolnoma odprtokodnega telefona"

V prejšnjih poglavjih smo pokazali, da mobilni telefoni niso le operacijski sistem in aplikacije, ampak preplet več slojev programske opreme, dobaviteljske verige in strojno-specifičnih komponent. Zato je pojem "popolnoma odprtokoden telefon" v praksi dvoumen: odvisen je od tega, ali govorimo o strogi (idealni) definiciji ali o uporabnem približku, ki še vedno omogoča vsakodnevno rabo. V tem poglavju najprej opredelimo, kaj bomo v nadaljevanju šteli kot "popolnoma" oziroma "pretežno" odprtokodno rešitev, nato pa postavimo kriterije po slojih sistema. Na koncu kriterije dopolnimo še s praktičnimi merili uporabnosti, saj odprtost brez vzdrževanja in funkcionalne naprave za uporabnika hitro postane zgolj akademska lastnost.

4.1 Definicija: idealni in praktični cilj

Če pojem "popolnoma odprtokoden" razumemo dobesedno, bi to pomenilo, da je **vsa** programska oprema, ki se izvaja na napravi (na glavnem procesorju in pomožnih podsistemih), na voljo kot prosta in odprtokodna programska oprema z možnostjo pregleda, sprememb in redistribucije [28, 19]. V kontekstu pametnih telefonov pa se skoraj vedno pojavijo vsaj tri trde omejitve:

(1) zaprt modemski podsistem, (2) zaprta strojna programska oprema za Wi-Fi/Bluetooth, kamero, grafični procesor in druge podsisteme, ter (3) mehanizmi preverjenega zagona, ki lahko omejijo nalaganje lastnih gradenj, če lastnik nima nadzora nad ključi oziroma je zagonski nalagalnik zaklenjen [11].

Zato v diplomski nalogi ločimo dva cilja:

Idealni model ("strogo popolnoma odprtokoden") Naprava je **brez binarnih dodatkov** (angl. *binary blobs*), vsi gonilniki in strojna programska oprema so FOSS, celoten sistem pa je možno zgraditi in preveriti iz izvorne kode (po možnosti s ponovljivimi gradnjami) [34]. Ta model uporabimo kot referenčni ideal, za katerega pričakujemo, da je danes pri mainstream telefonih praviloma nedosegljiv.

Praktični model ("pretežno odprtokoden") Večina programskega sklada (zagonska konfiguracija, jedro, uporabniški prostor, sistemske storitve in privzete aplikacije) je FOSS in realno vzdrževana, hkrati pa so neizogibno zaprte komponente (zlasti modem in določena strojna programska oprema) čim bolj omejene z arhitekturo, izolacijo in uporabniškim nadzorom. Ta model je naš operativni cilj za kasnejšo analizo in študijo primera.

Ključna ideja praktičnega modela je, da "odprtost" ni binarna, ampak večdimenzionalna: ne merimo le licence, temveč tudi možnost neodvisnega vzdrževanja, zamenljivost komponent, stopnjo izolacije zaprtih podsistemov in dejansko uporabnost naprave.

4.2 Slojni pogled na telefon

Za sistematično ocenjevanje potrebujemo slojni model, ki je skladen z arhitekturo iz slike 3.1. V nadaljevanju uporabljamo naslednje sloje:

1. **Zagonska veriga** (angl. *boot chain*): od začetnega zagona do nalaganja jedra in uporabniškega prostora.

2. **Jedro in gonilniki**: jedro Linux (ali prilagojeno jedro) ter gonilniki v jedru oziroma uporabniškem prostoru.

3. **Strojna programska oprema**: za podsisteme (modem, Wi-Fi/Bluetooth, grafični procesor, kamera, senzorji ipd.).

4. **Sistemske storitve in vmesni sloj** (angl. *middleware*): npr. sistemske storitve in abstrakcijska plast strojne opreme (HAL) v Androidu oziroma klasične Linux storitve v uporabniškem prostoru.

5. **Privzete aplikacije in distribucija aplikacij**: osnovni nabor aplikacij ter način nameščanja in posodabljanja dodatnih aplikacij.

Vsak sloj ocenjujemo v dveh dimenzijah: (1) **odprtost** in (2) **praktična uporabnost**. Odprtost brez uporabnosti je za telefon kot primarno napravo pogosto neuporabna, uporabnost brez odprtosti pa ne odgovori na raziskovalno vprašanje naloge.

4.3 Kriteriji odprtosti po slojih

V tej sekciji postavimo merila, s katerimi bomo kasneje primerjali konkretne rešitve. Kriteriji so namenoma zapisani tako, da so preverljivi (ali je nekaj objavljeno, zgrajeno, vzdrževano, zamenljivo), ne pa zgolj deklarativni.

4.3.1 Zagonska veriga

Pri zagonski verigi nas zanima predvsem, ali ima lastnik naprave realno možnost naložiti lasten sistem, ne da bi bil popolnoma odvisen od proizvajalca. Kriteriji:

- **Odklep zagonskega nalagalnika**: ali je uradno podprt in dokumentiran, ter ali ne zahteva nepovratnih korakov, ki uporabnika trajno omejijo.

- **Preverjen zagon in lastni ključi**: ali je možno ohraniti koncept preverjenega zagona, hkrati pa uporabljati lastne podpise (ravnotežje med integriteto in lastništvom) [11].
- **Dokumentiranost namestitve**: ali je postopek namestitve alternativnega sistema javno opisan in reproducibilen (ne le neuraden postopek brez garancij).

4.3.2 Jedro in gonilniki

Pri jedru je ključno vprašanje dolgoročno vzdrževanje. Če je strojna podpora del glavne razvojne veje jedra, je možnost dolgotrajnih varnostnih posodobitev bistveno večja [38]. Kriteriji:

- **Podpora v glavni veji**: ali so ključni gonilniki vključeni v uradno jedro ali pa gre za dobaviteljsko izpeljanko (angl. *fork*).
- **Odprtost gonilnikov**: ali so gonilniki FOSS ali pa so prisotni binarni dodatki v jedru oziroma uporabniškem prostoru.
- **Vzdrževalni signal**: ali je vidna aktivnost posodobitev, varnostnih popravkov in vključevanja sprememb v izvorni projekt (angl. *upstream*).

4.3.3 Strojna programska oprema

Strojna programska oprema je tipično največja ovira za strogo odprt telefon. V praksi zato ločimo "popolno odprtost" od "omejevanja škode", kjer zaprt podsistem poskušamo izolirati. Kriteriji:

- **Razpoložljivost izvorne kode**: ali je strojna programska oprema odprtokodna; če ne, ali obstajajo odprte alternative (tudi če z omejitvami).
- **Obseg in privilegiji**: kateri podsistemi so zaprti in kako privilegirani so (modemski podsistem je praviloma bolj problematičen kot npr. strojni program posameznega senzorja).

- **Izolacija**: ali arhitektura zmanjšuje vpliv zaprtih komponent, saj ranljivosti v zaprtih skladih predstavljajo realno napadalno površino [24, 17].

4.3.4 Sistemske storitve in vmesni sloj

Pri Androidu je pomembna meja med odprtim jedrom platforme (AOSP) in dobaviteljskimi komponentami in implementacije HAL, pri mobilnem Linuxu pa, ali je uporabniški prostor sestavljen iz standardnih paketov distribucije. Kriteriji:

- **Delež FOSS komponent**: koliko ključnih sistemskih storitev je odprtokodnih in zamenljivih.
- **Odvisnost od lastniških storitev**: ali sistem za osnovno delovanje zahteva centralizirane, zaprtokodne storitve ali pa ima odprte alternative.
- **Ponovljive gradnje**: ali je realno mogoče neodvisno preveriti binarne izdaje, vsaj za kritične dele sistema [34].

4.3.5 Privzete aplikacije in distribucija aplikacij

Na zadnjem sloju se pokaže "vsakodnevnost" naprave: tudi zelo odprt sistem je za mnoge uporabnike neuporaben, če nima realnega načina nameščanja aplikacij ali če mora uporabnik takoj poseči po zaprtih ekosistemih. Kriteriji:

- **Privzete aplikacije**: ali so osnovne aplikacije FOSS in brez nepotrebne telemetrije.
- **Trgovina in repozitoriji**: ali je primarni kanal namestitve aplikacij odprt (npr. paketni repozitorij ali trgovina *(*F-Droid)), ter ali podpira preverljive posodobitve.

- **Neodvisnost od proizvajalca**: ali uporabnik lahko vzdržuje sistem in aplikacije tudi po tem, ko proizvajalec naprave uradno zaključi podporo.

Sloj	Kaj preverjamo
Zagonska veriga	odklep zagonskega nalagalnika; možnost lastnih podpisov; dokumentiran postopek namestitve
Jedro in gonilniki	podpora v glavni veji; odprti gonilniki; aktivno vzdrževanje
Strojna programska oprema	odprta koda ali alternative; obseg zaprtih podsistemov; izolacija
Sistemske storitve	delež FOSS; odvisnost od lastniških storitev; (kjer je možno) ponovljive gradnje
Aplikacije in distribucija	FOSS privzete aplikacije; odprti repozitoriji; neodvisnost od proizvajalca

Tabela 4.1: Povzetek slojev in tipičnih kriterijev odprtosti v predlaganem modelu.

4.4 Kriteriji praktične uporabnosti

Ker je ciljna naprava telefon, mora sistem pokriti vsaj osnovne funkcionalnosti, sicer primerjava postane nepoštena (npr. odprt sistem brez delujočih klicev je "odprt", vendar ne deluje kot telefon). V tej nalogi praktično uporabnost ocenjujemo skozi naslednje sklope:

- **Osnovne funkcije telefona**: klici, SMS, mobilni podatki, stabilnost povezave.
- **Ključna strojna podpora**: kamera (vsaj uporabna kakovost), GPS, Bluetooth, Wi-Fi, zvok, senzorji; po potrebi tudi NFC.

- **Avtonomija in zmogljivost**: poraba energije v mirovanju in pri rabi, segrevanje, odzivnost.
- **Ekosistem aplikacij**: ali obstaja realen nabor aplikacij za tipične potrebe uporabnika; posebej problematične so aplikacije, ki uporabljajo preverjanje integritete [4].
- **Posodobitve in vzdrževanje**: kako pogosto prihajajo varnostne posodobitve in kako enostavno jih je namestiti (posodobitve po zraku, angl. *over-the-air*, OTA; paketni upravljalnik; ročne posodobitve).

4.5 Način ocenjevanja: primerjalni rezultat

Za kasnejšo analizo projektov in študijo primera potrebujemo konsistenten način primerjave. Namesto absolutne ocene "je / ni odprtokoden" uvedemo preprost primerjalni rezultat, kjer za vsak sloj ocenimo: (1) odprtost in (2) uporabnost. V praksi bomo uporabljali tri stopenjske vrednosti (0–2), kjer: 0 pomeni "pretežno zaprto ali neuporabno", 1 pomeni "mešano / kompromis", 2 pa "pretežno odprto ali zrelo za rabo".

Ker so sloji različno pomembni, uporabimo uteži. Jedro in možnost posodabljanja sta običajno bolj kritična za življenjsko dobo naprave kot npr. to, ali je privzeta aplikacija za koledar FOSS (če jo lahko zamenjamo). Uteži v tej fazi postavimo kot izhodišče, kasneje pa jih pri diskusiji po potrebi utemeljimo s kontekstom študije primera:

Skupni rezultat nato interpretiramo predvsem primerjalno (katera rešitev je bližje praktičnemu idealu), ne kot absolutno "certifikacijo". Tak pristop nam omogoča, da v naslednjih poglavjih bolj transparentno pokažemo, kje posamezna rešitev izgublja odprtost (npr. v strojni programski opremi) in kje izgublja uporabnost (npr. kamera ali avtonomija).

Sloj	Utež	Razlog (na kratko)
Zagonska veriga	0.15	lastništvo in možnost namestitve lastnega sistema
Jedro in gonilniki	0.30	vzdrževanje, varnostni popravki, potencial glavne veje
Strojna programska oprema	0.25	največji vir zaprtosti in napadalne površine
Sistemske storitve	0.20	odvisnosti, zamenljivost, stabilnost platforme
Aplikacije in distribucija	0.10	zamenljivo, a pomembno za vsakodnevno rabo

Tabela 4.2: Predlagane uteži slojev za primerjalno oceno v nadaljevanju naloge.

4.6 Povzetek

V tem poglavju smo postavili model "popolnoma odprtokodnega telefona", ki loči idealni (strogi) cilj od praktičnega (pretežno odprtega) cilja. Telefon smo razdelili na sloje in za vsak sloj določili preverljive kriterije odprtosti ter merila praktične uporabnosti. Model bomo v nadaljevanju uporabili kot okvir za analizo obstoječih odprtokodnih mobilnih rešitev in za empirično študijo primera, kjer bo pomembno predvsem to, kateri kompromisi so neizogibni in kateri so posledica konkretnih arhitekturnih ali organizacijskih odločitev projekta.

Literatura

[1] Abbas Acar in sod. "50 Shades of Support: A Device-Centric Analysis of Android Security Updates". V: *Proceedings of the Network and Distributed System Security Symposium (NDSS) 2024*. Velik empirični pregled uradnih zapisov posodobitev (2014–2023); pokaže razlike med OEM-i, zamude in omejeno trajanje varnostne podpore pri delu naprav. 2024. DOI: 10.14722/ndss.2024.24175. URL: https://www.ndss-symposium.org/wp-content/uploads/2024-175-paper.pdf (pridobljeno 14. 1. 2026).

[2] Android. *Android Enterprise Recommended Requirements*. Zahteve programa vključujejo tudi "Support current shipping release + one major OS upgrade" in zahteve glede varnostnih posodobitev. n.d. URL: https://www.android.com/enterprise/recommended/requirements/ (pridobljeno 14. 1. 2026).

[3] Android Developers. *Migration overview (Google Play Games Services)*. Vsebuje razlago, da je GMS Core/Google Play Services Googlov lastniški sloj na Androidu. Nov. 2025. URL: https://developer.android.com/games/pgs/migration_overview (pridobljeno 14. 1. 2026).

[4] Android Developers. *Overview of the Play Integrity API*. Okt. 2025. URL: https://developer.android.com/google/play/integrity/overview (pridobljeno 13. 1. 2026).

[5] Android Developers. *SafetyNet (deprecated): The SafetyNet Attestation API is deprecated and has been replaced by the Play Integrity API*. Opozorilo na strani navaja zamenjavo SafetyNet Attestation API s

Play Integrity API. Jan. 2026. URL: https://developer.android.com/privacy-and-security/safetynet (pridobljeno 13. 1. 2026).

[6] Android Open Source Project. *Android shared system image*. Opis Project Treble: ločitev strojno-specifičnega *vendor* dela in generičnega OS dela ter vmesnik VINTF. Dec. 2025. URL: https://source.android.com/docs/core/architecture/partitions/shared-system-image (pridobljeno 14. 1. 2026).

[7] Android Open Source Project. *Android Upgrade Invite*. Stran eksplicitno navaja, da Google Mobile Services (GMS) ni del AOSP in je na voljo le z licenco. Dec. 2025. URL: https://source.android.com/docs/core/ota/upgrade_invite (pridobljeno 14. 1. 2026).

[8] Android Open Source Project. *AOSP overview*. Opis Androida kot odprtokodnega programskega sklada za različne tipe naprav. 2025. URL: https://source.android.com/docs/setup/about (pridobljeno 19. 11. 2025).

[9] Android Open Source Project. *Interface versioning*. Opisuje mejo system.img/vendor.img pri Treble in zahtevo po verzioniranih vmesnikih. Avg. 2024. URL: https://source.android.com/docs/core/architecture/hidl/versioning (pridobljeno 14. 1. 2026).

[10] Android Open Source Project. *Mainline (modular system components)*. Opis modularnih posodobitev (Mainline) prek Google Play system update ali OTA mehanizmov partnerjev. Dec. 2025. URL: https://source.android.com/docs/core/ota/modular-system (pridobljeno 14. 1. 2026).

[11] Android Open Source Project. *Verified Boot*. Dokumentacija mehanizma Android Verified Boot (AVB). 2024. URL: https://source.android.com/docs/security/features/verifiedboot (pridobljeno 16. 2. 2026).

[12] Apple. *App code signing process in iOS, iPadOS, tvOS, watchOS, and visionOS*. Apple Platform Security / Security Guide. Dec. 2024. URL:

https://support.apple.com/guide/security/app-code-signing-process-sec7c917bf14/web (pridobljeno 14. 1. 2026).

[13] Apple Developer. *App Review Guidelines*. Nov. 2025. URL: https://developer.apple.com/app-store/review/guidelines/ (pridobljeno 14. 1. 2026).

[14] François Beaufort. *Media updates in Chrome 62*. Chrome for Developers; razlaga Widevine L1/L3 in povezavo z TEE. Sep. 2017. URL: https://developer.chrome.com/blog/media-updates-in-chrome-62 (pridobljeno 14. 1. 2026).

[15] Bitmovin. *Widevine Security Levels in Depth*. Sekundarni industrijski vir; opisuje tipično povezavo L1 → HD/UHD, L3 → SD. n.d. URL: https://developer.bitmovin.com/playback/docs/widevine-security-levels-in-web-video-playback (pridobljeno 14. 1. 2026).

[16] Conner Bradley. "Sustainable Security: Exploring Longevity Challenges and Solutions for IoT". V razdelku o dolgoživosti avtor izrecno omenja, da projekti, kot je LineageOS, podaljšujejo programsko podporo za naprave brez uradne podpore, npr. z portanjem novejših izdaj Androida na starejše telefone. Master's thesis. Ottawa, Ontario: Carleton University, 2023. URL: https://www.cisl.carleton.ca/~cbradley/data/papers/MastersThesis.pdf (pridobljeno 15. 1. 2026).

[17] Andrew Davis. "Cellular Baseband Security". Bachelor's thesis. Georgia Institute of Technology, 2012. URL: https://repository.gatech.edu/bitstreams/8cb73e2d-184d-41a3-8069-1d21d48ccc9a/download (pridobljeno 14. 1. 2026).

[18] e Foundation. */e/OS is a complete, fully "deGoogled", mobile ecosystem*. 2026. URL: https://e.foundation/e-os/ (pridobljeno 13. 1. 2026).

[19] Free Software Foundation. *What is Free Software?* Opredelitev proste programske opreme in štiri temeljne svoboščine. Jan. 2026. URL: https://www.gnu.org/philosophy/free-sw.html (pridobljeno 16. 2. 2026).

[20] Google for Developers. *Widevine DRM: Overview*. Okt. 2024. URL: https://developers.google.com/widevine/drm/overview (pridobljeno 14. 1. 2026).

[21] Google Play Console Help. *Use the Play Integrity API to detect risky interactions and fight abuse*. Uradna dokumentacija za razvijalce v Play Console; opisuje signale/verdict (npr. "genuine and certified Android device"). n.d. URL: https://support.google.com/googleplay/android-developer/answer/11395166?hl=en (pridobljeno 15. 1. 2026).

[22] GrapheneOS Project. *GrapheneOS: the private and secure mobile OS*. Glej razdelek "About". 2026. URL: https://grapheneos.org/ (pridobljeno 13. 1. 2026).

[23] Halium Project. *Halium – Halium documentation*. Uradni porting guide; opiše Halium kot projekt za poenotenje HAL za GNU/Linux sisteme na napravah s prednameščenim Androidom. 2023. URL: https://docs.halium.org/ (pridobljeno 14. 1. 2026).

[24] Grant Hernandez in sod. "FIRMWIRE: Transparent Dynamic Analysis for Cellular Baseband Firmware". V: *Network and Distributed System Security (NDSS) Symposium 2022*. 2022. DOI: 10.14722/ndss.2022.23136. URL: https://www.cise.ufl.edu/~butler/pubs/ndss22-firmwire.pdf (pridobljeno 14. 1. 2026).

[25] libhybris contributors. *libhybris: Load Android drivers from regular GNU/Linux processes*. README: libhybris omogoča nalaganje Android (bionic) knjižnic/gonilnikov znotraj procesov, ki uporabljajo npr. glibc ali musl. 2025. URL: https://github.com/libhybris/libhybris (pridobljeno 14. 1. 2026).

[26] LineageOS Project. *LineageOS – LineageOS Android Distribution*. Opis na strani navaja: "A free and open-source operating system for various devices, based on the Android mobile platform." URL: https://lineageos.org/ (pridobljeno 13. 1. 2026).

[27] Iliyan Malchev, Amith Dsouza in Veerendra Bhora. *Treble Plus One Equals Four*. Razlaga, kako arhitektura okoli Treble vpliva na stroške nadgradenj in podporne časovnice pri čipovjih (SoC) ter OEM-ih. Dec. 2020. URL: https://android-developers.googleblog.com/2020/12/treble-plus-one-equals-four.html (pridobljeno 14. 1. 2026).

[28] Open Source Initiative. *The Open Source Definition*. Uradna definicija odprtokodne programske opreme (OSI). Feb. 2024. URL: https://opensource.org/osd (pridobljeno 16. 2. 2026).

[29] PINE64. *PinePhone*. Opis naprave PinePhone, poudari podporo za mainline Linux, strojna stikala za zasebnost ter osnovne specifikacije. n.d. URL: https://pine64.org/devices/pinephone/ (pridobljeno 14. 1. 2026).

[30] PINE64 Wiki. *PinePhone*. Dokumentacija in kontekst (izdaje, stanje programske opreme, namestitev, specifikacije, OS-ji). Nov. 2024. URL: https://wiki.pine64.org/wiki/PinePhone (pridobljeno 14. 1. 2026).

[31] postmarketOS Project. *About postmarketOS*. Opis ciljev in arhitekture postmarketOS kot razširitve distribucije Alpine Linux na pametne telefone. 2024. URL: https://wiki.postmarketos.org/wiki/About_postmarketOS (pridobljeno 19. 11. 2025).

[32] Purism. *Librem 5*. Uradna stran naprave Librem 5 (cilji, zasebnost, osnovne lastnosti in usmeritev v odprtokodnost). n.d. URL: https://puri.sm/products/librem-5/ (pridobljeno 14. 1. 2026).

[33] Purism Documentation. *Librem 5 – Documentation*. Tehnična dokumentacija za Librem 5 (programski sklad, uporaba, komponente). n.d. URL: https://docs.puri.sm/Librem_5/ (pridobljeno 14. 1. 2026).

[34] Reproducible Builds. *Reproducible Builds*. Projekt in smernice za reproducibilne gradnje programskih paketov. n.d. URL: https://reproducible-builds.org/ (pridobljeno 16. 2. 2026).

[35] Sailfish OS Documentation. *Hardware Adaptation Development Kit – libhybris*. Opiše uporabo libhybris kot most med GNU libc uporabniškim prostorom in Android (bionic) komponentami pri strojnih prilagoditvah. 2024. URL: https://docs.sailfishos.org/Tools/Hardware_Adaptation_Development_Kit/ (pridobljeno 14. 1. 2026).

[36] StatCounter GlobalStats. *Mobile Operating System Market Share Worldwide*. Dostopno prek StatCounter GlobalStats, podatki za obdobje 2024–2025. 2025. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (pridobljeno 13. 1. 2026).

[37] Team Counterpoint. *Global Smartphone Sales Share by Operating System*. Counterpoint Research, Insight. Nov. 2025. URL: https://counterpointresearch.com/en/insights/global-smartphone-os-market-share (pridobljeno 13. 1. 2026).

[38] The Linux Kernel Archives. *The Linux Kernel Archives*. Uradna stran izdaj jedra Linux in referenca za "mainline". n.d. URL: https://www.kernel.org/ (pridobljeno 16. 2. 2026).

[39] Ken Thompson. "Reflections on Trusting Trust". V: *Communications of the ACM* 27.8 (1984), str. 761–763. DOI: 10.1145/358198.358210.

[40] UBports Community. *Convergence*. Uradna UBports dokumentacija; razlaga koncept konvergence (telefon ↔ namizje) in cilje uporabniške izkušnje. 2026. URL: https://docs.ubports.com/en/latest/humanguide/other-design-considerations/convergence.html (pridobljeno 15. 1. 2026).

[41] UBports Community. *Introduction – UBports documentation*. Uradna dokumentacija projekta UBports/Ubuntu Touch; opis ciljev projekta in kratka zgodovina. 2026. URL: https://docs.ubports.com/en/latest/about/introduction.html (pridobljeno 14. 1. 2026).

[42] Widevine Help. *How to determine if Android device is security Level 1 or Level 3*. Google Support; vsebuje tudi status DEVICE_IS_PROVISIONED_SD_ONLY

za Widevine L3. n.d. URL: https://support.google.com/widevine/answer/6072714?hl=en (pridobljeno 14. 1. 2026).