Sigurnost ugrađenih sustava

Bezi, Josip

Master's thesis / Diplomski rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:953025

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Josip Bezi

SIGURNOST UGRAĐENIH SUSTAVA

DIPLOMSKI RAD

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Josip Bezi

Matični broj: 0016110465

Studij: Baze podataka i baze znanja

SIGURNOST UGRAĐENIH SUSTAVA

DIPLOMSKI RAD

Mentor:

doc. dr. sc. Igor Tomičić

Varaždin, veljača 2025.

Josip Bezi

Izjava o izvornosti

Izjavljujem da je ovaj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Sigurnost ugrađenih sustava je postala ključna tema u području kibernetičke sigurnosti, s obzirom na sve veću prisutnost povezanih uređaja i rastući broj prijetnji koje ciljaju hardwareske komponente. Ovaj rad istražuje metode analize i zaštite ugrađenih sustava, s posebnim naglaskom na penetracijsko testiranje, analizu firmware-a i napade bočnim kanalima.Istraživanje je pokazalo da su mnogi ugrađeni sustavi podložni ranjivostima zbog nedovoljno implementiranih sigurnosnih mjera. Posebno su analizirani napadi koji koriste UART, JTAG i SPI sučelja za neovlašteni pristup i manipulaciju firmware-om, kao i metode koje koriste elektromagnetske emisije i potrošnju energije za ekstrakciju osjetljivih podataka.U radu su također predstavljene strategije obrane, uključujući siguran boot, obfuskaciju hardwareskog dizajna i korištenje fizički neklonirajućih funkcija (PUF-ova) za autentifikaciju uređaja. Zaključeno je da je integracija sigurnosnih mjera u ranim fazama dizajna ključna za otpornost hardwareskih sustava na napade.S obzirom na rast IoT uređaja i njihovu široku primjenu u industriji, zdravstvu i kritičnoj infrastrukturi, sigurnost hardwareskih komponenti ostaje jedan od najvažnijih izazova. Buduće istraživanje trebalo bi se usmjeriti na razvoj metoda za detekciju hardwareskih trojanaca, poboljšanje zaštite od napada bočnim kanalima te unapređenje sigurnosnih arhitektura ugrađenih sustava.

Ključne riječi: hardwareska sigurnost; penetracijsko testiranje; napadi bočnim kanalima; JTAG; UART; SPI; SWD; siguran boot; analiza firmware-a; IoT sigurnost; ugrađeni sustavi; fizički sloj sigurnosti

Sadržaj

1. Uvod

U svijetu koji je sve više povezan, sigurnost ugrađenih sustava postaje sve veća briga. Proliferacija uređaja Interneta stvari (IoT) ,od pametnih kućanskih aparata do industrijskih kontrolih sustava, proširila je površinu napada dostupnu zlonamjernim akterima. Budući da ovi uređaji često kontroliraju kritične funkcije i rukuju osjetljivim podacima, njihova je sigurnost sastavni dio ukupnog krajolika kibernetičke sigurnosti [1]. Sigurnost hardware-a obuhvaća zaštitu ovih fizičkih uređaja od neovlaštenog pristupa i osigurava njihovu pouzdanost.

Ne može se precijeniti važnost hardwareske sigurnosti. S razvojem kibernetičkih prijetnji, tradicionalne sigurnosne mjere, koje su temeljene na software-u, više nisu dovoljne. Napadači sada ciljaju na hardwareski sloj, tj. iskorištavaju ranjivosti koje mogu dovesti do ozbiljnih posljedica kao što su povrede podataka, kvarove sustava, pa čak i fizičke štete [2]. Ovaj diplomski rad istražuje područje sigurnosti hardware-a, pružajući teorijske temelje i praktične uvide u osiguranje hardwareskih komponenti od raznih napada i prikazuju praktične metodologije napada.

Primarni je izazov sve veća ranjivost hardwareskih sustava pred sofisticiranim kibernetičkim napadima. Dok su softwareske sigurnosne mjere značajno napredovale, hardwareska je sigurnost i dalje relativno nerazvijena i često zanemarena [3].

Ciljevi ovog rada su sljedeći :

- Pružiti detaljan pregled računalne sigurnosti, s fokusom na povijesni kontekst i primjere sigurnosnih proboja iz stvarnog svijeta;
- Ispitati osnove IoT-a i pametnih uređaja, ističući njihove sigurnosne izazove i ranjivosti;
- Proučiti teoriju hardwareske sigurnosti, raspravljajući o metodologijama napada, komunikacijskim protokolima.

Opseg ovog rada uključuje istraživanje načela sigurnosti hardware-a, analizu različitih vektora napada i prezentaciju primjera iz stvarnog svijeta i studija slučaja za ilustraciju praktičnih implikacija ovih napada. Iako su praktični napadi izvan opsega ovog odjeljka, oni će biti detaljno obrađeni u posebnom poglavlju.

2. Internet stvari i tipovi ugrađenih sustava

Internet stvari (eng. Internet of things - IoT) označava mrežu fizičkih objekata, odnosno "stvari", koji sadrže senzore, software i druge tehnologije koje omogućavaju umrežavanje i razmjenu podataka s drugim uređajima i sistemima putem interneta. Takvi uređaji mogu biti od običnih kućanskih aparata do industrijskih alata. Primarni cilj IoT-a jest proširenje internetske povezanosti van klasičnih uređaja, kao što su računala i pametni mobiteli, na niz fizičkih uređaja i svakodnevnih predmeta koji nemaju pristup internetu [1].

Putem interneta stvari ti se objekti mogu čitati i kontrolirati iz daljine kroz postojeću mrežnu infrastrukturu. Ovakva integracija poboljšava efikasnost, točnost kao ekonomičnost uz dodanu vrijednost na smanjenje ljudske prisutnosti. Glavne komponente IoT sistema uključuju :

1. **Senzori i uređaji** : To su komponente koji prikupljanju podatke iz stvarnog svijeta. Ti podaci mogu biti jednostavni kao što je temperatura ili razina vlage do kompleksnijih podataka kao što je strujanje videa i zvuka;
2. **Povezanost** : Prikupljeni podaci se moraju na neki način poslati drugdje. Generalno su korištene razne bežične mreže kao što je Bluetooth, Wi-Fi i mobilne mreže;
3. **Obrada podataka** : Nakon što su podaci preneseni na neku pohranu, software ima ulogu obradu istih;
4. **Korisničko sučelje** : Prikupljeni i obrađeni podaci su potrebni biti u formatu koji je čitljiv krajnjim korisnicima. Tu se odnosi na obavijesti, web sučelje ili mobilna aplikacija koja prikazuje podatke.

2.1. Tipovi i značajke pametnih uređaja

Pametni uređaji su podskup IoT-a. Dizajnirani su na način da poboljšaju korisničko iskustvo s integracijom naprednih značajki preko interneta. Neki od tipova takvih uređaja uključuju :

- **"Smart Home" uređaji** : termostati, svjetlosni sistemi, kamere, brave i ostale stvari koje upravljaju kućanstvom;
- **Nosivi predmeti** : uređaji kao što su pametni satovi i uređaji za zdravstvene metrike;
- **Pametni kućni uređaji** : predmeti poput pametnih hladnjaka, pećnica i perilica rublja koji nude napredne značajke i povezanost;
- **Povezana vozila** : automobili opremljeni pristupom internetu i senzorima za pružanje navigacije u stvarnom vremenu i ažuriranja održavanja;

Slika 1: Primjer sustava za praćenje podataka na biljkama; preuzeto iz [4]

- **Industrijski IoT uređaji** : koriste se u proizvodnji i upravljanju opskrbnim lancem za nadzor i kontrolu industrijskih procesa;
- **Zdravstveni uređaji** : medicinski uređaji koji prate vitalne znakove pacijenata i prenose podatke pružateljima zdravstvenih usluga.

Uz prije navedene uređaje, ruteri i Android TV-i su među najzastupljenijim pametnim uređajima današnjice, koji omogućuju korisnicima pristup internetu i zabavnim sadržajima. Njihova široka primjena i popularnost čine ih ključnim komponentama modernog digitalnog doma. Kasnije u radu će se neki od navedenih uređaja koristiti za analizu sigurnosti.

2.2. Prijetnje sigurnosti Interneta stvari i stvarni napadi

IoT obuhvaća sve uređaje koji se mogu povezati na internet, od pametnih telefona do kućanskih aparata. Ogromna količina i raznolikost "stvari" koje čine IoT sadrže značajnu količinu korisničkih podataka, koji mogu biti meta cyber kriminalaca. Povećan broj povezanih uređaja stvara više prilika za kompromitiranje sigurnosti. Posljedice proboja sigurnosti IoT-a mogu biti izuzetno štetne jer utječu na virtualne i fizičke sustave. Prema Kaspersky [5] ovo su izazovi sigurnosti interneta :

- **Nedostatak testiranja i razvoja** : Mnogi proizvođači IoT uređaja stavljaju sigurnost u drugi plan u žurbi da lansiraju proizvode na tržište, što može rezultirati sigurnosnim rizicima i nedostatkom ažuriranja;
- **Default lozinke** : Mnogi IoT uređaji dolaze s unaprijed postavljenim slabim lozinkama koje korisnici često ne mijenjaju, što ih čini podložnima napadima brute-force;

- **IoT malware i ransomware** : Porast povezanih uređaja povećava rizik od malware-a i ransomware-a, posebno botnet malware-a koji su često viđeni u IoT okruženju;
- **Zabrinutost za privatnost podataka** : IoT uređaji prikupljaju, prenose i obrađuju ogromne količine podataka koji se mogu dijeliti ili prodavati trećim stranama bez potpunog razumijevanja korisnika;
- **Inficirani IoT uređaji kao baza za napade** : Uređaji mogu biti korišteni za distribuciju napada odbijanja usluge (DDoS), što može utjecati na organizacije, ali i pametne domove;
- **Nesigurna sučelja** : Česti problemi uključuju slabu ili nepostojeću enkripciju i nedovoljnu autentifikaciju podataka;
- **Porast rada na daljinu** : Pandemija Covid-19 povećala je rad na daljinu, što je naglasilo sigurnosne ranjivosti IoT uređaja koji su korišteni izvan sigurnih mreža organizacija;
- **Kompleksna okruženja** : Prosječno kućanstvo ima više povezanih uređaja, gdje jedna sigurnosna greška može ugroziti cijelu mrežu.

U moderno vrijeme sve su učestaliji napadi na računalne sustave. U vrlo kratkom vremenu se može napraviti šteta mnogobrojnim korisnicima. U nastavku su navedena četiri stvarna napada koja prikazuju bitnost zaštite računalnih sustava:

- **Mirai botnet napad (2016)** : Stotine tisuća kompromitiranih uređaja su uključeni u botnet Mirai, što je rezultiralo privremenim padom usluga kao što su Spotify, Netflix i PayPal;
- **VPNFilter malware (2018)** : VPNFilter malware je zarazio preko pola milijuna routera u više od 50 zemalja, prikupljajući informacije, blokirajući mrežni promet i kradući lozinke;
- **Hakiranje Tesla Model X (2020)** : Stručnjak za kibernetičku sigurnost hakirao je Tesla Model X u manje od dvije minute koristeći ranjivost Bluetooth-a;
- **Hakiranje kamera Verkada (2021)** : Švicarski hakeri kompromitirali su 150,000 live kamera tvrtke Verkada, nadzirući aktivnosti u javnim zgradama i privatnim organizacijama.

3. Teorija i metodologije

3.1. Uvod u teorijske osnove hardwareske sigurnosti

Hardwareska sigurnost postaje sve važniji aspekt informacijske sigurnosti zbog rastuće sofisticiranosti napada koji ciljaju ugrađene sustave i IoT uređaje. Napadači sve više koriste slabosti u fizičkom sloju sustava, zaobilazeći tradicionalne softwareske zaštite. Cilj ovog poglavlja je predstaviti teorijske osnove hardwareske sigurnosti te metodologije za zaštitu i testiranje ranjivosti hardwareskih uređaja.

Postoje više vrsta prijetnji , neke od kojih su opisani u nastavku. Napadi bočnim kanalima (Side-Channel Attacks - SCA) se izvršavaju putem analize elektromagnetskog zračenja, potrošnje energije i vremena izvršavanja radi izvlačenja povjerljivih informacija. Hardwareski trojanci (Hardware Trojans - HTs) su manipulirani sklopovi ubačeni u proizvodni proces koji omogućuju backdoor pristup ili onesposobljavanje sustava. Napadi na fizički sloj je fizičko spajanje na interne komunikacijske magistrale uređaja (UART, SPI, JTAG) za preuzimanje ili modificiranje firmware-a.

3.2. Metodologije sigurnosne analize hardware-a

Hardwareska sigurnost zahtijeva multidisciplinarni pristup koji kombinira analizu dizajna, implementacije i testiranja sigurnosnih ranjivosti. Ovaj odjeljak razmatra ključne metodologije koje se koriste u istraživanju i praktičnoj primjeni hardwareske sigurnosti.

3.2.1. Penetracijsko testiranje hardware-a

Penetracijsko testiranje (pentesting) je ključni postupak za identificiranje slabosti u hardwareskim sustavima. Knjiga Practical Hardware Pentesting [6] opisuje sljedeće korake:

- **Prikupljanje informacija** : identifikacija i analiza uređaja, konektora i komunikacijskih protokola;
- **Analiza firmware-a** : ekstrakcija i dekompilacija software-a ugrađenog u uređaj radi traženja sigurnosnih propusta;
- **Napadi na fizičke sučelja** : pristup putem UART, SPI i JTAG konektora kako bi se zaobišli sigurnosni mehanizmi;
- **Bočni napadi** : praćenje potrošnje energije, elektromagnetskih emisija ili kašnjenja u izvođenju instrukcija radi izvlačenja tajnih podataka.

3.2.2. Analiza napada bočnim kanalima

Napadi bočnim kanalima omogućuju napadačima da iskoriste nenamjerne informacije koje odaju uređaji. The Hardware Hacking Handbook [7] detaljno opisuje kako:

- Analiza potrošnje energije može otkriti obrasce šifriranja i omogućiti prepoznavanje kriptografskih ključeva;
- Elektromagnetska emisija može omogućiti napadačima praćenje operacija procesora iz daljine;
- Vrijeme izvršavanja instrukcija može otkriti podatke o ključevima ako su implementacije kriptografskih algoritama nepravilno zaštićene.

Jedna od poznatih metoda napada bočnim kanalima je Differential Power Analysis (DPA), koja uspoređuje male varijacije u potrošnji energije kako bi izvukla kriptografske ključeve.

3.2.3. Analiza hardwareskih trojanaca

Hardwareski trojanci predstavljaju jednu od najvećih prijetnji integriranih sklopova. Ove manipulacije mogu onesposobiti ili modificirati funkcionalnost čipa, stvoriti skrivene backdoor pristupe i izazvati degradaciju performansi sustava.

Metode detekcije uključuju **analizu električnih karakteristika čipa** (promjene u potrošnji energije i frekvencijskim karakteristikama mogu ukazivati na prisutnost trojanca) te **strukturne analize sklopova** (korištenje mikroskopa i rendgenskih snimaka za identifikaciju neautoriziranih promjena u dizajnu čipa).

3.3. Zaštitne metodologije i protumjere

Kako bi se umanjila ranjivost hardwareskih sustava, razvijene su brojne obrambene strategije.

3.3.1. Sigurno projektiranje hardware-a

Sigurnost se mora ugrađivati u fazi dizajna hardware-a. Machine Learning-Enhanced Analysis and Design for Trustworthy Integrated Circuits [8] opisuje nekoliko ključnih metoda. Jedna od metoda je dodavanje sigurnosnih mehanizama na silicijskoj razini, a to su kriptografski moduli i sigurni boot mehanizmi. Druga je metoda uporaba Hardwareskog autentifikatora (PUFs - Physically Unclonable Functions), odnosno jedinstvene električne karakteristike čipa koriste se za autentifikaciju uređaja.

3.3.2. Tehnike otkrivanja manipulacija

Zaštita od napada uključuje metode detekcije i analize neovlaštenih promjena u hardwareskim sustavima. Jedna od metoda je praćenje varijacija u naponskoj potrošnji (otkrivanje bočnih napada i umetnutih trojanaca). Nadalje, putem medote analize vremenskog ponašanja čipa mogu se uočiti promjene u kašnjenjima koje ukazuju na prisutnost skrivenog koda u procesoru. Metodom hardwareskog šifriranja osjetljivih podataka smanjuje se vjerojatnost uspjeha napada bočnim kanalima.

3.3.3. Praktične implementacije zaštitnih mjera

Moderni sustavi koriste različite pristupe za osiguranje hardware-a. Kod korištenja OTP memorije (One-Time Programmable Memory) kriptografski ključevi, pohranjeni u nepromjenjivoj memoriji, štite od napada na firmware. Putem Secure Boota, provjera potpisa firmware-a osigurava da uređaj učitava samo ovjereni kod. Nadalje, obfuskacija hardwareskih dizajna smanjuje mogućnosti napada reverznim inženjeringom.

4. Tehnologije i alati

U ovom odjeljku će biti detaljan pregled hardwareskih i softwareskih komponenti korištenih u razvoju i realizaciji ovog rada. Rasprava će obuhvatiti specifične alate i tehnologije koje se koriste, uključujući fizičke uređaje i softwareske platforme koje su omogućile uspješnu implementaciju projekta. Ispitivanjem ovih elemenata.

4.1. Software

4.1.1. Visual Studio Code

Visual Studio Code (VS Code) je besplatan, jednostavan i moćan uređivač koda koji je razvio Microsoft. Dizajniran je za programere svih razina, nudi širok spektar značajki i podržava različite programske jezike. Njegova fleksibilnost i mogućnost proširenja čine ga jednim od najpopularnijih IDE-a danas.

Jedne od ključnih prednosti VS Code-a su njegova brzina i lagano korištenje. Za razliku od drugih integriranih razvojnih okruženja (IDE), VS Code koristi malo sistemskih resursa, čime omogućuje brzo pokretanje i glatko izvođenje, čak i na slabijim računalima. Unatoč svojoj maloj veličini, nudi moćne funkcije kao što su sintaksno isticanje, automatsko dovršavanje koda (IntelliSense), podrška za debugging, kontrola verzija (Git) te integracija s brojnim alatima i proširenjima [9].

VS Code također podržava više od 150 programskih jezika pa programeri mogu raditi u istom okruženju bez potrebe za dodatnim alatima. IntelliSense tehnologija olakšava kodiranje tako da pruža automatsko dovršavanje i kontekstualne prijedloge na temelju sintakse jezika. Integrirani alat za debuggiranje pruža jednostavno pronalaženje i ispravljanje grešaka bez napuštanja uređivača.

Još jedna ključna značajka je mogućnost proširenja putem ekstenzija. Kroz Visual Studio Marketplace, korisnici mogu dodavati ekstenzije za različite jezike, okvire (frameworks), alate za produktivnost i integraciju s cloud servisima. Ova modularnost omogućuje prilagodbu uređivača prema potrebama korisnika, bilo da se radi o web razvoju, programiranju u Pythonu, Javi, C++, ili čak radu s bazama podataka.

4.1.2. Putty

PuTTY je besplatan i open-source terminalski emulator koji podržava mrežne protokole kao što su SSH (Secure Shell), Telnet, Rlogin i serijske veze. Razvio ga je Simon Tatham za Windows operativni sustav, no danas je dostupan i na drugim platformama. Njegova je primarna svrha omogućiti sigurnu daljinsku administraciju poslužitelja, čineći ga jednim od najčešće korištenih alata u domeni mrežnog i sustavskog inženjeringa [10].

PuTTY nudi podršku za više mrežnih protokola, opisanih u nastavku. SSH je siguran

protokol za udaljeni pristup koji koristi enkripciju kako bi zaštitio podatke dok Telnet je protokol za udaljeno povezivanje, no bez enkripcijske zaštite. Rlogin je alternativa Telnetu koja omogućuje autentifikaciju putem korisničkog računa. Također, postoje serijske veze koje daju mogućnost povezivanja s uređajima putem serijskog porta, što je korisno za konfiguraciju mrežne opreme i drugih ugrađenih sustava.

4.1.3. WSL

Windows Subsystem for Linux (WSL) predstavlja značajku operacijskog sustava Windows 10 i Windows 11 koja omogućuje pokretanje Linux okruženja unutar Windows operativnog sustava, bez potrebe za virtualnim strojevima ili dual-boot konfiguracijama. Ovaj podsustav omogućuje korištenje Linux distribucija i njihovih komandnih alata, čime se značajno poboljšava kompatibilnost i interoperabilnost između Windows i Linux sustava [11].

WSL pruža programerima i administratorima izravno pokretanje Linux aplikacija i skripti iz Windows okruženja, što kombinira prednosti oba sustava. Korisnici mogu istovremeno koristiti Windows aplikacije i Linux terminal, pristupati datotekama između sustava i koristiti Windows upravitelje datoteka za rad s Linux datotekama.

WSL podržava instalaciju različitih Linux distribucija, kao što su:

- Ubuntu
- Debian
- Kali Linux
- Arch Linux
- openSUSE
- Fedora

Distribucije se mogu instalirati izravno putem Microsoft Storea, čime se olakšava pristup raznim Linux alatima.

4.1.4. Arduino studio

Arduino IDE (Integrated Development Environment) predstavlja službeno razvojno okruženje za programiranje Arduino mikrokontrolera. To je open-source alat koji omogućuje korisnicima pisanje, kompajliranje i učitavanje koda na Arduino ploče putem jednostavnog i intuitivnog grafičkog sučelja. Arduino IDE je dostupan na operativnim sustavima Windows, macOS i Linux, a njegova jednostavnost i široka zajednica korisnika čine ga jednim od najpopularnijih alata za razvoj ugrađenih sustava [12].

Jedna od glavnih prednosti Arduino IDE-a je jednostavnost korištenja. Sučelje se sastoji od:

- Editora koda – omogućuje pisanje i uređivanje programa sa sintaksnim isticanjem;
- Konzole za poruke – prikazuje informacije o kompajliranju i greškama;
- Ugrađenog serijskog monitora – omogućuje komunikaciju s mikrokontrolerom putem UART (Universal Asynchronous Receiver-Transmitter) protokola;
- Jednostavne komande za učitavanje i pokretanje programa – korisnici mogu jednim klikom kompajlirati i prenijeti kod na mikrokontroler.

Arduino IDE koristi C i C++ jezike s dodatnim pojednostavljenim funkcijama specifičnim za Arduino platformu. Standardne knjižnice posjeduju jednostavnu implementaciju složenih funkcionalnosti, npr. upravljanje senzorima, komunikacija s vanjskim modulima i rad s mrežnim protokolima.

4.1.5. Logic 2

Saleae Logic 2 je napredni software za analizu digitalnih signala koji se koristi zajedno s Saleae Logic analizatorima. Ovaj alat pruža snimanje, dekodiranje i analizu digitalnih i analognih signala u stvarnom vremenu. Time se olakšava razvoj i otklanjanje pogrešaka u embedded sustavima, IoT uređajima i digitalnim elektroničkim krugovima. Namijenjen je inženjerima, programerima i istraživačima koji rade s serijskim protokolima, mikrokontrolerima i niskonaponskim digitalnim sklopovima.

Saleae Logic 2 pruža istovremeno snimanje i analizu digitalnih i analognih signala, ovisno o modelu analizatora. Glavne značajke uključuju podršku za do 16 digitalnih kanala (ovisno o modelu analizatora), visoku brzinu uzorkovanja do 500 MS/s (milijuna uzoraka u sekundi) te dekodiranje i vizualizaciju analognih signala s preciznim prikazom naponskih razina.

Jedna od najvažnijih funkcija Saleae Logic 2 software-a je mogućnost automatskog dekodiranja serijskih komunikacijskih protokola, što značajno olakšava otklanjanje grešaka u embedded sustavima. Software podržava:

- I^2C (Inter-Integrated Circuit)
- SPI (Serial Peripheral Interface)
- UART/RS-232
- 1-Wire
- CAN (Controller Area Network)
- USB (Universal Serial Bus)
- Ethernet i druge industrijske protokole

Dekodirani podaci se prikazuju u obliku čitljivog teksta, čime se stvara lakša interpretacija i analiza.

Saleae Logic 2 koristi Python skripte za automatizaciju analizu podataka i izvođenje složenih obrada signala. Ova funkcionalnost dopušta korisnicima da prilagode dekodiranje podataka specifično za svoj projekt, automatiziraju analizu rezultata koristeći prilagođene algoritme i integriraju Saleae Logic 2 s drugim alatima unutar razvojnih okruženja.

Software nudi intuitivno sučelje s mogućnostima navedima u nastavku:

- povećavanje i smanjivanje pregleda signala radi lakše analize,
- sinkronizacija digitalnih i analognih podataka za usporedbu signala,
- dodavanje markera i bilješki kako bi se istaknuli ključni događaji,
- vizualizacija omogućuje bolje razumijevanje ponašanja signala u realnom vremenu, što je korisno za otkrivanje anomalija i optimizaciju sustava.

Software podržava izvoz podataka u različitim formatima (*CSV, VCD, MATLAB*), čime se postiže jednostavna analiza u drugim alatima. Osim toga, korisnici mogu dijeliti snimljene podatke s timovima putem cloud integracija, što je posebno korisno u kolaborativnim okruženjima.

4.1.6. OpenOCD

OpenOCD (Open On-Chip Debugger) je open-source alat namijenjen debuggiranju, programiranju i testiranju mikrokontrolera i ugrađenih sustava. Primarno se koristi u embedded razvoju, gdje osigurava niski pristup hardware-u. Podržava JTAG, SWD (Serial Wire Debug), i Boundary Scan tehnike te olakšava real-time analizu koda. OpenOCD služi kao posrednik između GDB debuggiranja, hardwareskih programatora i ciljanog mikrokontrolera, što dovodi do učinkovitog testiranja i razvoja firmware-a [13].

OpenOCD podržava širok spektar ARM, RISC-V, MIPS, x86 i drugih arhitektura, što ga čini univerzalnim alatom za razvoj ugrađenih sustava. Kompatibilan je s mikrokontrolerima poznatih proizvođača kao što su:

- STMicroelectronics (STM32)
- Texas Instruments (MSP430, Sitara)
- NXP (LPC serija)
- ESP32 (Espressif)
- Microchip (SAM serija)

Ova široka podrška se može razvijati za različite platforme koristeći isti alat.

OpenOCD radi kao posrednik između GDB-a (GNU Debugger) i ciljanog mikrokontrolera. Omogućava postavljanje breakpointa, inspekciju registara i memorije, kao i pokretanje i

zaustavljanje koda u realnom vremenu. Ova funkcionalnost je ključna za ispitivanje i uklanjanje grešaka u ugrađenim sustavima.

OpenOCD podržava nekoliko metoda pristupa hardware-u. JTAG (Joint Test Action Group) se koristi za hardwaresko testiranje i programiranje mikrokontrolera. SWD (Serial Wire Debug) je alternativa JTAG-u, specifična za ARM Cortex-M mikrokontrolere, te pruža brži i efikasniji debugging. Boundary-Scan obavlja testiranje i verifikaciju PCB-a bez pokretanja firmware-a.

Korištenjem OpenOCD-a, programeri mogu izravno čitati i pisati memoriju mikrokontrolera, upravljati registrima procesora te pristupati perifernim uređajima bez potrebe za pokretanjem operativnog sustava.

OpenOCD podržava semihosting. Zbog toga je moguće izvršavanje dijela koda na host računalu, dok mikrokontroler izvršava druge dijelove programa. Ova značajka stvara jednostavniju interakciju između ciljanog sustava i razvojnog računala.

Jedna od ključnih funkcionalnosti OpenOCD-a je mogućnost programiranja ugrađene Flash memorije na mikrokontrolerima. Podržava različite vrste memorije, kao što su NOR Flash, NAND Flash i SPI Flash. To osigurava učinkovito programiranje firmware-a izravno s računala.

4.1.7. UrJTAG

UrJTAG je open-source alat za rad s JTAG (Joint Test Action Group) sučeljem, namijenjen testiranju, programiranju i otklanjanju pogrešaka u digitalnim sklopovima. Koristi se prvenstveno u ugrađenim sustavima, PCB testiranju i programiranju flash memorije. Njegova je glavna svrha stvoriti univerzalni alat za interakciju s JTAG uređajima bez potrebe za vlasničkim software-om.

UrJTAG je nasljednik starijeg projekta JTAG Tools, a poboljšan je dodavanjem povećane kompatibilnosti s različitim hardwareskim platformama i bolje podrške za različite arhitekture.

UrJTAG omogućuje detekciju, interakciju i upravljanje JTAG uređajima, uključujući:

- Mikrokontrolere (ARM, AVR, MIPS, x86, itd.),
- FPGA čipove (Xilinx, Altera, Lattice),
- Procesorske jezgre u embedded sustavima,
- Serijske Flash memorije povezane preko JTAG-a.

Ova široka podrška čini ga pogodnim za niskonaponsku dijagnostiku i testiranje ugrađenih sustava.

UrJTAG pruža programiranje SPI, NOR i NAND flash memorija koje su povezane putem JTAG sučelja. To ga čini korisnim alatom za firmware nadogradnje, popravak nefunkcionalnih uređaja i razvoj ugrađenih sustava.

Jedna od ključnih funkcija JTAG standarda je Boundary-Scan testiranje koje korisnicima daje automatizirano ispitivanje sklopa i pronalazak grešaka. Ono izvršava provjeru ispravnosti lemljenih komponenti na PCB-u, dijagnostiku problema s vezama između čipova i analizu signala na digitalnim linijama bez potrebe za osciloskopom.

UrJTAG nudi interaktivno naredbeno sučelje (CLI) gdje je skriptiranje izvedeno pomoću Tcl i shell skripti. Ova funkcionalnost je korisna za automatizirane testove, proizvodne procese i analizu digitalnih signala.

UrJTAG podržava razne JTAG adaptere i debuggere, kao što su:

- FTDI FT2232
- Parallel Port JTAG dongle
- Amontec JTAGkey
- USB-Blaster (za Altera FPGA-e)
- DirtyJTAG

Ova kompatibilnost pruža prilagodbu različitim radnim okruženjima bez potrebe za skupim vlasničkim alatima.

Budući da je UrJTAG open-source projekt, aktivno ga održava zajednica programera i inženjera. Korisnici mogu prilagoditi kod vlastitim potrebama, što je korisno za industrijske, istraživačke i obrazovne primjene.

4.2. Hardware

4.2.1. Arduino

Arduino je open-source hardwareska i softwareska platforma dizajnirana za razvoj elektroničkih i embedded sustava. Ova platforma nudi jednostavan razvoj interaktivnih projekata korištenjem mikrokontrolerskih ploča koje su kompatibilne sa širokim spektrom senzora, aktuatora i komunikacijskih modula. Arduino je izuzetno popularan u obrazovanju, IoT projektima, automatizaciji i eksperimentalnoj elektronici, zahvaljujući svojoj jednostavnosti i fleksibilnosti [14].

Glavna komponenta svake Arduino ploče je mikrokontroler, koji upravlja unosima i izlazima podataka. Najčešće korišteni modeli su:

- Arduino Uno – temelji se na ATmega328P mikrokontroleru;
- Arduino Mega – koristi ATmega2560, s većim brojem I/O pinova;
- Arduino Nano – minijaturna verzija s istim mikrokontrolerom kao Uno;

- Arduino Due – temelji se na 32-bitnom ARM Cortex-M3 procesoru;
- Arduino MKR serija – specijalizirane ploče za IoT aplikacije s podrškom za WiFi i Bluetooth.

Arduino ploče imaju različit broj digitalnih i analognih I/O pinova, zbog kojih je moguće povezivanje s vanjskim komponentama. U nastavku su navedene neke vrste pinova:

- Digitalni I/O pinovi – koriste se za upravljanje LED diodama, prekidačima i drugim komponentama;
- PWM (Pulse Width Modulation) pinovi – omogućuju simulaciju analognih izlaza pomoću digitalnih signala;
- Analogni ulazni pinovi – omogućuju očitavanje signala iz senzora i potenciometara;
- Komunikacijski pinovi – podržavaju SPI, I²C i UART protokole za povezivanje s vanjskim uređajima.

Većina Arduino ploča radi na 5V ili 3.3V, ovisno o modelu. Standardne vrijednosti napajanja su ulazni napon (7-12V preporučeno, 6-20V maksimalno), radni napon mikrokontrolera (5V – Uno, Mega ili 3.3V – MKR, Due) i struja po I/O pinu (do 20mA po pinu, s maksimalnim opterećenjem cijelog mikrokontrolera do 200mA).

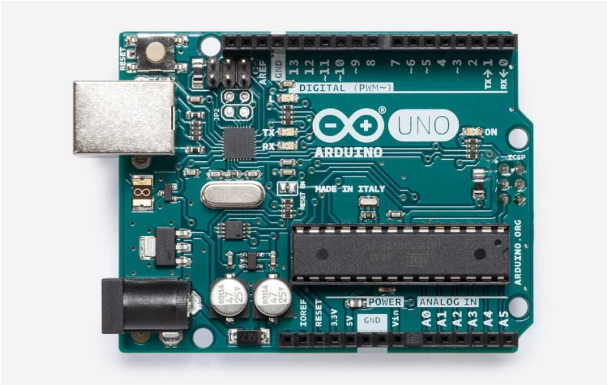
Mikrokontroleri na Arduino pločama imaju ugrađenu memoriju za spremanje programa i podataka. Flash memorija se koristi se za pohranu programskog koda (16 KB – 256 KB). SRAM je privremena memorija za varijable (2 KB – 8 KB). EEPROM je nehlapljiva memorija za trajno spremanje podataka (1 KB – 4 KB).

Također, Arduino podržava različite metode komunikacije. Serijska komunikacija (UART) koristi USB ili TX/RX pinove za komunikaciju s računalom ili drugim uređajima. I²C i SPI omogućuju povezivanje s vanjskim senzorima, ekranima i drugim mikrokontrolerima. Komunikacija putem WiFi-a i Bluetooth-a je dostupna na pločama poput Arduino MKR WiFi 1010 i Arduino Nano 33 BLE.

Arduino se programira pomoću Arduino IDE-a ili Arduino CLI-a, koristeći pojednostavljeni C/C++ jezik. Kod se prenosi na mikrokontroler putem USB konekcije, a debugging je moguć pomoću serijskog monitora.

4.2.2. Raspberry Pi i Raspberry Pi Pico

Raspberry Pi i Raspberry Pi Pico predstavljaju dva različita, ali međusobno povezana hardwareska rješenja koja nudi Raspberry Pi Foundation. Raspberry Pi je punopravno jednopločno računalo (SBC - Single Board Computer) koje može pokretati operacijske sustave poput Linuxa. Raspberry Pi Pico je mikrokontrolerska ploča namijenjena za embedded sustave, senzorske mreže i real-time aplikacije. Obje platforme koriste se u obrazovanju, istraživanju, industrijskoj automatizaciji i IoT projektima [15].



Slika 2: Arduino Uno; preuzeto iz [14]

4.2.2.1. Raspberry Pi

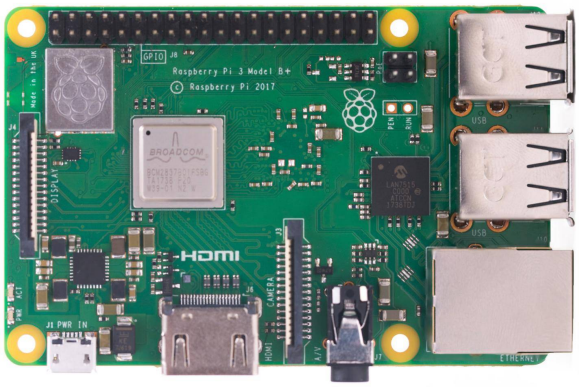
Raspberry Pi je kompaktno i pristupačno jednopločno računalo koje nudi razvoj aplikacija poput edge computinga, IoT uređaja, multimedijskih sustava i automatizacije.

Hardwareske specifikacije (Raspberry Pi 3 Model B) :

- Procesor: 64-bitni četverojezgreni ARM Cortex-A53 @ 1.2 GHz,
- Radna memorija (RAM): 1GB LPDDR2 SDRAM,
- Grafika: VideoCore IV GPU, podrška za 1080p video dekodiranje,
- Pohrana: microSD kartica za OS i podatke,
- Povezivost:
  - 1x HDMI.
  - 4× USB 2.0.
  - Fast Ethernet (10/100 Mbps).
  - Wi-Fi 802.11n i Bluetooth 4.1.
- GPIO (General Purpose Input/Output) pinovi: 40-pin header za povezivanje senzora i dodatnih modula.

U nastavku su navedene glavne značajke Raspberry Pi računala:

- Mogućnost pokretanja punopravnog operacijskog sustava – Raspberry Pi može koristiti Raspberry Pi OS (bivši Raspbian), kao i razne Linux distribucije poput Ubuntua, a podržava i Windows IoT Core.
- Široka podrška za periferne uređaje – zahvaljujući USB, HDMI, GPIO, I²C, SPI i UART portovima, moguće je povezati širok spektar vanjskih uređaja.
- Podrška za IoT i Edge Computing – Raspberry Pi dozvoljava implementaciju lokalnih servera, MQTT sustava, AI aplikacija i automatizacije.



Slika 3: Raspberry Pi 3B; preuzeto iz [16]

- Razvojni i edukacijski alat – koristi se u školama i fakultetima za učenje programiranja, elektronike i računalne znanosti.

4.2.2.2. Raspberry Pi Pico

Za razliku od Raspberry Pi SBC-a, Raspberry Pi Pico je mikrokontroler, optimiziran za embedded aplikacije i real-time sustave. Opremljen je vlastitim RP2040 mikrokontrolerskim čipom koji nudi visok stupanj fleksibilnosti za niskonaponske projekte.

Hardwareske specifikacije (Raspberry Pi Pico):

- Mikrokontroler: RP2040, razvijen od strane Raspberry Pi Foundation,
- Procesor: Dvostruka ARM Cortex-M0+ jezgra, radnog takta do 133 MHz,
- Memorija:
  - 264KB SRAM,
  - 2MB QSPI Flash memorije (može se proširiti),
- Povezivost:
  - Ne podržava Wi-Fi i Bluetooth (osim verzije Pico W),
  - USB 1.1 (host i device mod),
- I/O mogućnosti:
  - 26 GPIO pinova s podrškom za PWM,
  - 3 ADC ulaza za očitavanje analognih signala,
  - Podrška za SPI, I²C i UART.

Glavne značajke Raspberry Pi Pico mikrokontrolera su:

- Niska potrošnja energije – pogodan za IoT senzore, baterijski pogonjene uređaje i real-time aplikacije;

- Podrška za MicroPython i C/C++ – nudi jednostavno programiranje u MicroPythonu ili C/C++ okruženju;
- Integracija s vanjskim senzorima – putem GPIO, PWM i ADC pinova stvoren je rad s različitim komponentama poput senzora temperature, svjetla i tlaka;
- Pico W verzija s Wi-Fi modulom – pruža bežičnu komunikaciju i IoT aplikacije.

4.2.3. FT232RL UART na USB adapter

FT232RL je USB na TTL serijski konverter koji stvara komunikaciju između računala i mikrokontrolera putem USB sučelja. Razvio ga je FTDI (Future Technology Devices International Ltd.), a koristi se za serijsku komunikaciju (UART) između računala i različitih uređaja koji koriste TTL (Transistor-Transistor Logic) naponske razine. FT232RL je široko korišten u razvoju embedded sustava, industrijskim aplikacijama i programiranju mikrokontrolera [17].

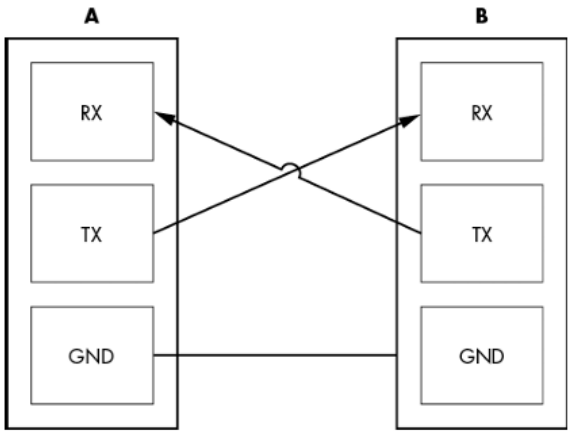
Glavne značajke FT232RL čipa su:

- Pretvorba USB signala u serijsku komunikaciju - FT232RL prijenosi podataka između računala i uređaja s TTL UART sučeljem. Ovaj čip pruža lako povezivanje mikrokontrolera (npr. Arduino, ESP8266, STM32) s računalom bez potrebe za dodatnim serijskim portovima;
- Fleksibilna naponska kompatibilnost - FT232RL podržava rad na 3.3V i 5V, što ga čini kompatibilnim s većinom modernih mikrokontrolera i senzora;
- Integrirani EEPROM za konfiguraciju - Ugrađeni EEPROM dopušta spremanje specifičnih postavki uređaja, kao što su serijski broj, VID/PID (Vendor/Product ID), konfiguracija CBUS pinova, čime se ostvaruje prilagodba prema potrebama korisnika;
- Široka kompatibilnost s operativnim sustavima - FT232RL ima ugrađenu podršku za Windows, macOS i Linux, što jamči brzu instalaciju i kompatibilnost bez dodatnih drajvera na modernim OS-ima.
- USB napajanje - FT232RL može napajati vanjske uređaje putem USB-a, uz strujnu potporu do 500mA na 5V.

4.3. Serijski protokoli i sučelja

4.3.1. UART

UART (eng. *universal asynchronous reciver/transmitter*) je protokol za serijsko sučelje koje dozvoljava komunikaciju s uređajima. UART je jedan od najčešćih adaptacija protokola na uređajima. U nekim slučajevima, nazvan je kao *serial*, *RS-232* ili *TTL Serial*. Sastoji se od samo 3 žice : **Rx** - receiver, **Tx** - transmitter i **GND**. Prepoznatljiv je prilikom analize kontaktnih površina putem osciloskopa ili logic analyzera. Protokol je asinkron, što znači da sam protokol



Slika 4: UART konekcija između uređaja; preuzeto iz [7]

nema *clock speed*, tj. obje stranke se moraju dogovoriti za brzinu prilikom inicijalizacije komunikacije. Receiver i transmitter označavaju da je protokol obostran te obje stranke mogu primati i slati podatke [7].

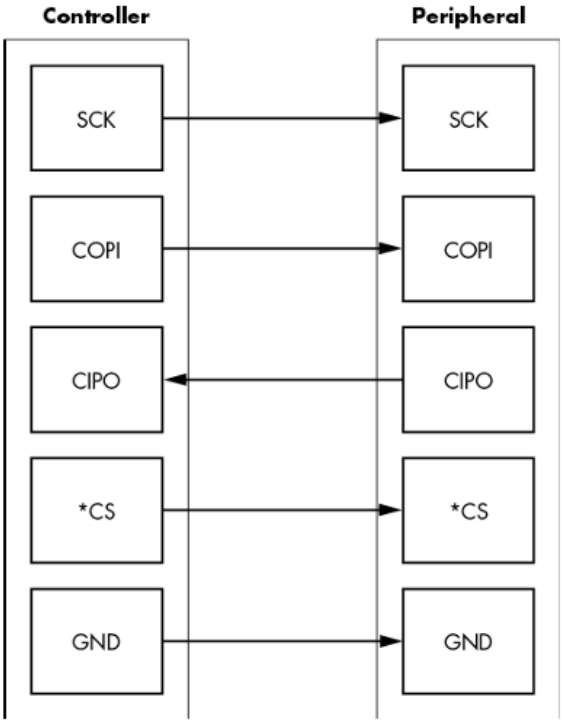
Žice se povezuju unakrsno (prijamnik jednog uređaja je odašiljač drugog i obrnuto). 4

4.3.2. SPI

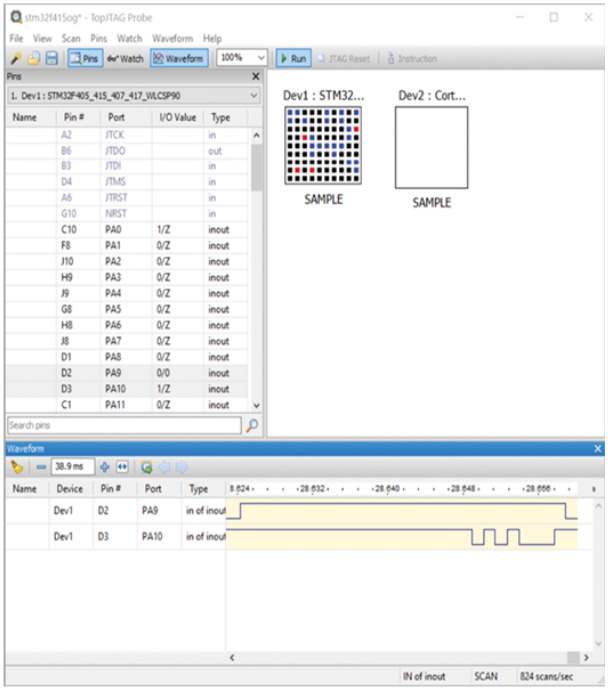
SPI (eng. *serial peripheral interface*) je serijsko sučelje za sinkronu komunikaciju. Generalno sadrži jedan kontroler i jedan ili više perifernih uređaja na sabirnici. Za razliku od UART-a, SPI je kontroler-periferno sučelje. To znači da periferni uređaji odgovaraju samo na zahtjev kontrolera, tj. ne mogu samostalno uspostaviti komunikaciju. Sinkrona komunikacija označava da kontroler određuje brzinu (*clock speed*) te se sastoji od 5 žica : **SCK** (serial clock), **COPI** (controller out peripheral in), **CIPO** (controller in peripheral out), **CS** (chip select) i **GND** (uzemljenje) [7]. Jedno je od najčešćih implementacija komunikacije s EEPROM-ima. Dijagram spajanja može se vidjeti na slici 5.

4.3.3. JTAG

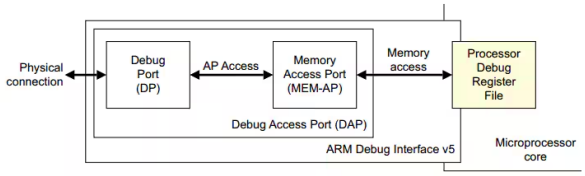
JTAG (eng. *Joint Test Action Group*) je hardwaresko debug sučelje i kritično je za sigurnost. JTAG standard je definiran pod oznakom IEEE 1149.1. Daje pristup unutarnjim registrima čipova bez potrebe za fizičkim sondama i kontaktima. JTAG koristi serijsko komunikacijsko sučelje koje se sastoji od nekoliko osnovnih signala : **TDI** (Test Data In), **TDO** (Test Data Out), **TCK** (Test Clock), **TMS** (Test Mode Select) i opcionalnog **TRST** (Test Reset). Jedna od glavnih primjena JTAG-a je **Boundary Scan** (slika 6), metoda koja pruža testiranje povezivosti između komponenti na tiskanoj pločici bez potrebe za fizičkim kontaktom. Također, JTAG se koristi za ISP (In-System Programming) mikrokontrolera, FPGA i drugih programabilnih uređaja te debugging i nadzor rada procesora [6].



Slika 5: SPI konekcija; preuzeto iz [7]



Slika 6: Boundary scan na BGA uređaju; preuzeto iz [7]



Slika 7: ARM Debug sučelje; preuzeto iz [18]

4.3.4. SWD

SWD (Serial Wire Debug) je serijski debug protokol razvijen od strane **ARM-a** kao alternativa **JTAG-u** za programiranje i ispravljanje softwareskih grešaka na mikroprocesorima i mikrokontrolerima. SWD koristi samo dvije žice, umjesto pet koliko koristi JTAG. To ga čini efikasnijim i pogodnijim za uređaje s ograničenim brojem pinova [18].

Osnovni signali SWD-a su SWDIO (Serial Wire Debug Input/Output), što je dvosmjerni podatkovni vod, i SWCLK (Serial Wire Clock), koji predstavlja taktni signal za sinkronizaciju prijenosa podataka.

Za razliku od JTAG-a, koji koristi sklop **TAP** (Test Access Port), SWD koristi Debug Access Port (**DAP**) koji omogućuje izravan pristup memoriji i registrima procesora (slika 7). Ovaj pristup olakšava **ISP** (In-System Programming) i dozvoljava debugging u stvarnom vremenu, uključujući postavljanje breakpoints-a, praćenje registara i očitavanje memorije bez prekida rada sustava.

5. Sigurnost u praksi

5.1. Traženje ranjivosti putem UART

Prvi uređaj koji će biti analiziran je Tenda N300 router. Tražimo ime modela putem interneta kako bi se saznale dodatne informacije o uređaju. Prilikom potrage pronađena je stranica FCC-a [19]. FCC (Federal Communications Commission) je nezavisna agencija Vlade Sjedinjenih Američkih Država. Svaki telekomunikacijski uređaj, koji ulazi u SAD, mora prvo proći kroz analizu FCC-a. FCC dostavlja svima dostupne slike i informacije o uređaju koje mogu pomoći prilikom slaganja vektora napada na pojedini uređaj.

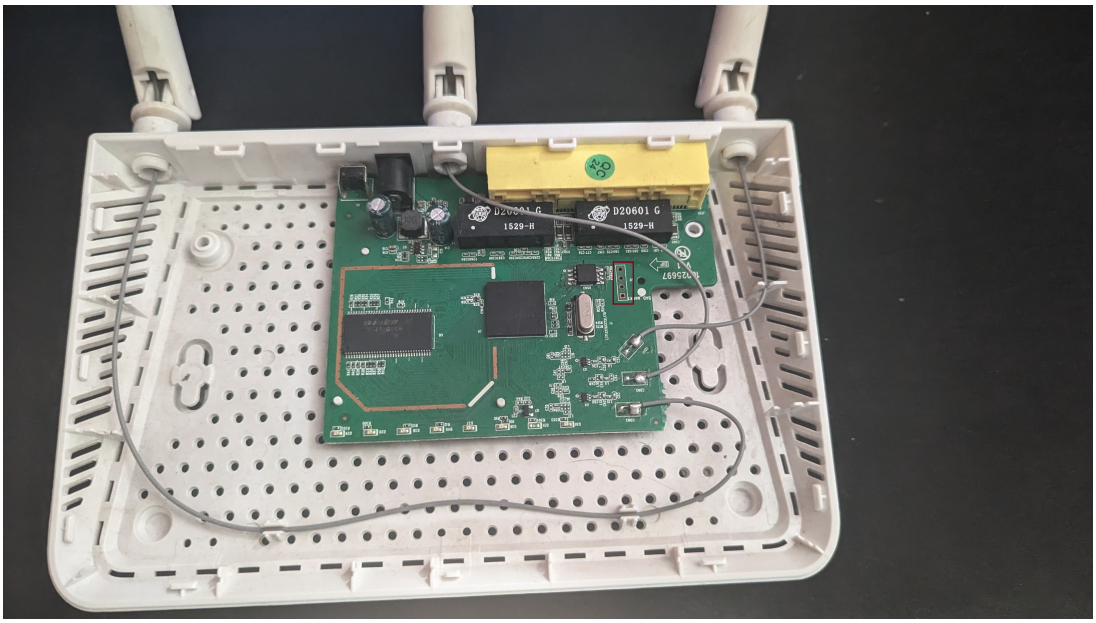
Nakon otvaranja uređaja, može se identificirati Broadcom BCM5357C0 procesor i KH25L1606E flash čip. Nadalje, pokraj flash čipa se nalaze 4 otvorene kontaktne površine označene crvenom bojom na slici 8. Multimetrom se ispituje GND, tj. jedan vod stavimo na poznati GND, kao što je metalni okvir gumba s gornje lijeve strane i svake otvorene kontaktne površine. Dalje, ispitujemo Vcc tako da se stavi jedan vod multimetra na GND i drugi na ostale površine. Dobiveni su sljedeći podatci vidljivi na tablici 1. Površine mogu označavati da je proizvođač ostavio otvorene debugging mogućnosti na tiskanoj pločici. Komunikacija se odvija putem nekih od prije objašnjenih protokola, kao što su UART ili JTAG. Spajamo kontakte na logic analyzer prema sljedećem dijagramu vidljivom na slici 9.

U nastavku, otvaramo program Logic2, spajamo logic analizer na PC preko USB-a i pokrećemo novu sesiju u programu. Uključujemo uređaj tako da ga samo priključimo na napon te pratimo izlaz na računalu. Prateći rezultat na Logic2 software-u, otkrivamo da kanal 1 (CH1, D0 u software-u) ima neki protok podataka (slika 10). Promatrajući rezultat, signal iz površina je nalik UART signalu pošto JTAG zahtijeva više prijenosnih pinova. Spajamo pinove na UART na USB adapter (slika 11). FT232RL se spaja na računalo te otvaramo putty kako bi se dobio terminal output na ekranu. „Serial line“ označava USB uređaj na računalu, a „Speed“ je tkz. Baud rate (broj signala u sekundi), gdje su 115200 i 9600 jedni od najzastupljenijih brzina 12. Nakon što se pokrenula sesija i uključio uređaj, dobiva se output na ekranu vidljiv u isječku 1.

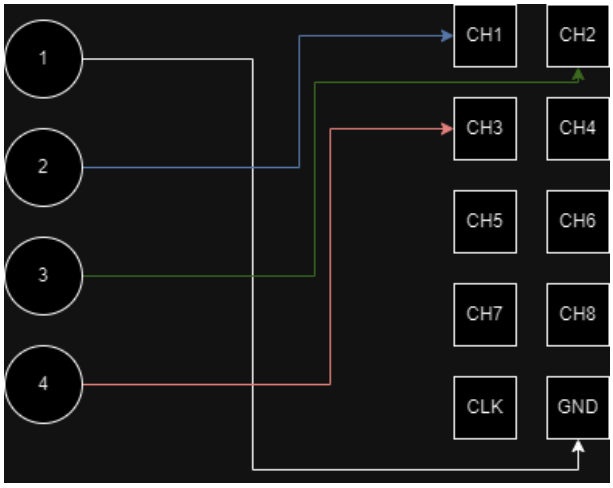
U prvim linijama output-a, primjećuje se da uređaj koristi CFE (Common Firmware Environment) bootloader, Id flash memorije, boot particije i datoteke (CFE mem, Data, BSS, Heap, Stack i Text) s početnim i krajnjim adresama te samom veličinom u bitovima. Nakon što se obavio proces dizanja OS-a na uređaju, davanjem signala putem UART-a na uređaj, pojavljuje se CLI (Command line interface) putem kojeg se mogu davati naredbe direktno OS-u. Unosom naredbe help prikazuju se opcije prikazane u isječku 2.

Kontaktna površina	Napon
1	GND
2	3.3V
3	3.3V
4	3.3V

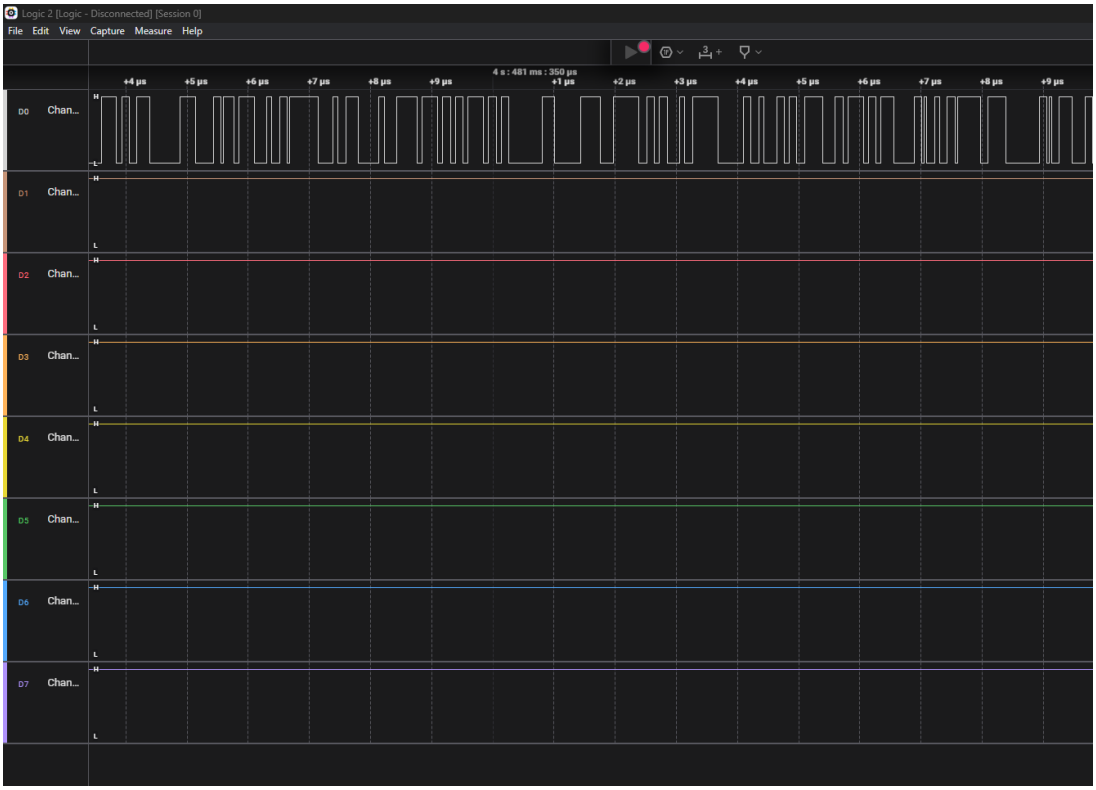
Tablica 1: Napon na kontaktnim površinama za Tenda N300



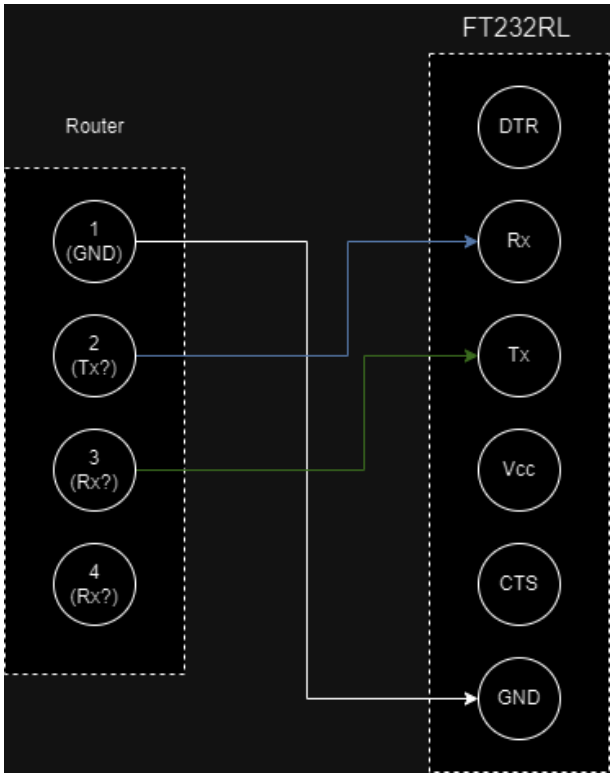
Slika 8: Tenda N300 router; autorska slika



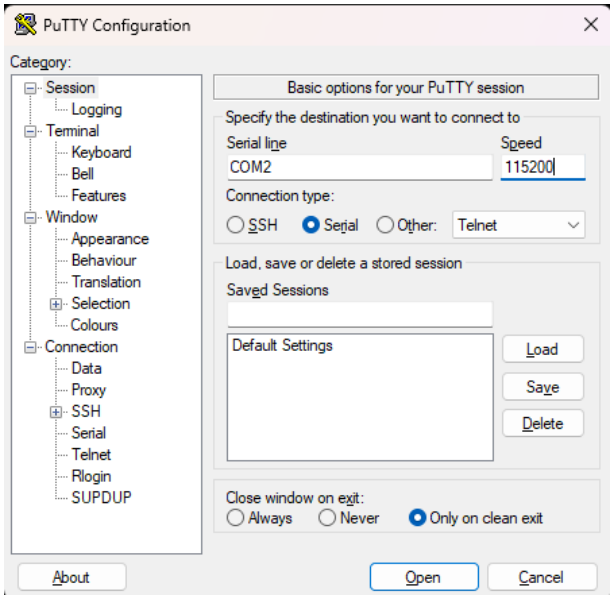
Slika 9: Dijagram spajanja na Logic analyser; autorska slika



Slika 10: Rezultat analiziranja UART-a; autorska slika



Slika 11: Spajanje na UART; autorska slika



Slika 12: Spajanje na Putty; autorska slika

```
loading...
Loader buf 80700000:0x73595754
LZMA Decompressing...done
System led init!


CFE version 5.100.138.22 based on BBP 1.0.37 for BCM947XX (32bit,SP,LE)
Build Date: 2015 06 08 - 06:50:01 CST (Zhuzeji@linux-bsp)
Copyright (C) 2000-2008 Broadcom Corporation.

Init Arena
Init Devs.
Boot partition size = 131072(0x20000)
get flash id : 0x14
Found an ST compatible serial flash with 32 64KB blocks; total size 2MB
get flash id : 0x14
get flash id : 0x14
disable EEE for all ports
ch->etc->chops->phywr at 8071d8b4
read reg test return 0x362
et0: Broadcom BCM47XX 10/100/1000 Mbps Ethernet Controller 5.100.138.22
CPU type 0x19749: 300MHz
Tot mem: 16384 KBytes

CFE mem:    0x80700000 - 0x807997A0 (628640)
Data:       0x8072F580 - 0x807327E0 (12896)
BSS:        0x807327E0 - 0x807337A0 (4032)
Heap:       0x807337A0 - 0x807977A0 (409600)
Stack:      0x807977A0 - 0x807997A0 (8192)
Text:       0x80700000 - 0x8072F578 (193912)
```

Isječak koda 1: Output putem UART-a na terminalu

```
CLI> help

reboot      restart     thread      time        nvram       syslog

debug       splx        mbuf        ifconfig    hqw         ping

ipnat       fw          route       envram      arp         tenda_arp

wl          wlconf      et
```

Isječak koda 2: Output putem UART-a na terminalu CLI

```
CFE> ^C
CFE> help
Available commands:

show clocks         Show current values of the clocks.
nvram               NVRAM utility.
reboot              Reboot.
flash               Update a flash memory device
batch               Load a batch file into memory and execute it
go                  Verify and boot OS image.
boot                Load an executable file into memory and execute it
load                Load an executable file into memory without
 ↪  executing it
save                Save a region of memory to a remote file via TFTP
ping                Ping a remote IP host.
arp                 Display or modify the ARP Table
ifconfig            Configure the Ethernet interface
help                Obtain help for CFE commands
```

Isječak koda 3: Output putem UART-a na terminalu CFE

Primjećujemo da je OS unix baziran i jako ograničen. S naredbama se mogu izvršavati jednostavne instrukcije kao što su ping i informacije o IP konfiguraciji. U daljnem procesu pokušavamo izbjeći OS, odnosno želimo raditi direktno sa CFE. Nakon resetiranja uređaja, u terminal dajemo SIGINT signal na način da se drže tipke CTRL i C prilikom boot-a kako bi se „ubilo“ dizanje OS-a. Poslije uspješnog gašenja procesa, pojavljuje se CFE CLI na kojemu se mogu davati instrukcije. Ponovnim unosom naredbe help, vide se instrukcije vidljive u isječku 3. Od navedenih, save se čini kao zanimljiva naredba pošto omogućava pohranu dijela memorije putem TFTP servera. Naredba save ima oblik vidljiv na isječku 4.

Varijabla „ip_adresa“ je odredišna adresa TFTP servera, a to je u ovom slučaju računalo na koji je spojen router. Varijabla „ime_datoteke“ je proizvoljno ime datoteke koju šaljemo, „po-

```
save ip_adresa:ime_datoteke početna_adresa dužina
```

Isječak koda 4: Save komanda

```
save 192.168.0.8:CFEMenu 0x80700000 628640
```

Isječak koda 5: Save komanda s ispunjenim parametrima

```
strings BSS > strings_bss.txt
```

Isječak koda 6: Strings komanda

četna_adresa“ označava početnu adresu datoteke u memoriji te „dužina“ je dužina u bitovima. Datoteke iz početnog output-a imaju sve navedene parametre što nudi jednostavan prijenos dokumenata. Zatim, router se spaja na računalo putem ethernet žice. Otvaramo Tftpd64 aplikaciju i pokrećemo TFTP server. Nakon što smo uključili router, otvaramo CFE CLI i pokrećemo sljedeću naredbu 5.

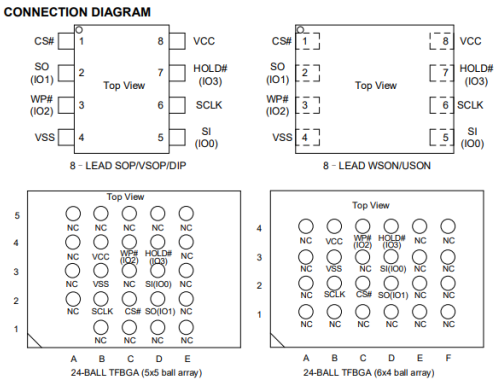
Na računalu se vidi prijenos datoteke. To je dokaz da je metoda prijenosa uspješna te se proces ponavlja i za ostale datoteke. Na kraju je pribavljeno 6 datoteka. Otvaranjem WSL, koji ima instaliranu verziju Ubuntu OS-a, radimo daljnju analizu datoteka. Nakon pokretanja naredbe file *, doznajemo da su sve datoteke tipa file, tj. neodređene su. Nadalje, pokušavamo izvući stringove iz tih datoteka kako bi dobili bolji dojam što smo dobili. Naredba strings omogućava pretraživanje svih ascii znakova s minimalno 4 znaka dužine. Dužina se može promijeniti zastavicom -min-len. Pokrećemo strings naredbu za svaku datoteku 6.

Naredbom pretražujemo sve stringove u datoteci te output pohranjujemo u posebnu datoteku. Tokom pretraživanja rezultata, možemo primijetiti da datoteka BSS sadrži neke konfiguracijske parametre koji su potrebni za inicijalizaciju routera. Jedan od tih parametra je „wl0_wpa_psk“ i njegova vrijednost „12345678“ , vidljivo na isječku 7. Ova vrijednost odgovara lozinki mreže koja je bila postavljena prije cijelog procesa sigurnosne analize.

Može se zaključiti da je nekom akteru potrebno kratko vrijeme s uređajem i osnovni alati za komunikaciju s hardware-om kako bi dobio password mreže. Zatim se može spojiti na mrežu i dalje tražiti ranjivosti na ostalim spojenim uređajima.

```
wl0.1_ifname=
wl0_wpa_psk=12345678
wait_time=20
```

Isječak koda 7: Strings outpun na BSS dokumentu



PIN DESCRIPTION

Pin Name	I/O	Description
CS#	I	Chip Select Input
SO (IO1)	I/O	Data Output (Data Input Output 1)
WP# (IO2)	I/O	Write Protect Input (Data Input Output 2)
VSS		Ground
SI (IO0)	I/O	Data Input (Data Input Output 0)
SCLK	I	Serial Clock Input
HOLD# (IO3)	I/O	Hold Input (Data Input Output 3)
VCC		Power Supply

Slika 13: Dokumentacija za GD25Q32C16 EEPROM; preuzeto iz [20]

5.2. Dohvaćanje firmware-a putem SPI protokola

5.2.1. TP-Link Router

Drugi uređaj koji će biti analiziran je router TP-link TL-WR841N. Otvaranjem uređaja možemo prepoznati par chipova kao što su Winbond W9425G6KH-5 DRAM memorija, Qualcomm QCA9533 procesor i 25Q32C16 flash chip. Tokom traženja informacija o flash chipu, nailazimo na sljedeću dokumentaciju vidljivu na slici 13. U ovome slučaju, Chip je 8-pin SOP (small outline package) chip. Chip ima implementirani SPI sučelje, što potvrđuje i sam dijagram pinova.

Za dohvaćanje podataka sa chipa koristit ćemo Arduino Nano kao uređaj za čitanje. Prvo se izrađuje Arduino sketch prema specifikacijama EEPROM-a, koji je vidljiv u isječku 8. Na početku skripte uključujemo SPI.h biblioteku koja sadrži funkcije za komunikaciju putem SPI protokola. Definiramo CS (Chip select) pin kao pin pod brojem 10 na arduinu. CS pin kontrolira koji je uređaj na SPI protoku aktivan (LOW omogućuje EEPROM komunikaciju dok HIGH onemogućuje).

U setup() funkciji prvo se postavlja komunikaciju arduina s računalom tako da definiramo Baud rate na 115200. Odabiremo EEPROM_CS pin kao izlaz gdje kontroliramo je li EEPROM aktivan prilikom SPI komunikacije. SPI.begin() inicijalizira SPI bus, a SPI.beginTransaction() započinje prijenos. Definiramo brzinu sata na 8MHz, što je dovoljno brzo jer EEPROM dopušta do 120MHz frekvencije. Određujemo MSBFIRST jer EEPROM prvo očekuje MSB (Most Significant Bit). Na kraju se definira SPI_MODE0 jer EEPROM dopušta SPI mode 0, odnosno SCK (System clock) je u stanju čekanja, a podaci se dohvaćaju na rising edge-u (CPOL i CPHA = 0).

```
#include <SPI.h>

#define EEPROM_CS 10

void setup() {
  Serial.begin(115200);
  pinMode(EEPROM_CS, OUTPUT);
  SPI.begin();

  SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));

  for (long i = 0; i < 4194304; i++) {
    byte data = readEEPROM(i);
    Serial.print(data, HEX);
    Serial.print(" ");
  }

  SPI.endTransaction();
}

void loop() {}

byte readEEPROM(long address) {
  byte rdata = 0xFF;
  digitalWrite(EEPROM_CS, LOW);
  SPI.transfer(0x03);
  SPI.transfer((address >> 16) & 0xFF);
  SPI.transfer((address >> 8) & 0xFF);
  SPI.transfer(address & 0xFF);
  rdata = SPI.transfer(0xFF);
  digitalWrite(EEPROM_CS, HIGH);
  return rdata;
}
```

Isječak koda 8: Strings outpun na BSS dokumentu

EEPROM pin	Arduino pin
S0	12
S1	11
SCLK	13
CS	10

Tablica 2: Dijagram spajanja SPI na arduino

U petlji iteriramo kroz svih 4MB (4194304 bajtova, tj. 32 Mbits iz specifikacije) i pozivamo funkciju readEEPROM kojoj prosljeđujemo adresu i. Rezultat pohranjujemo u varijablu data te ispisujemo izlaz na serial monitor. Na kraju završavamo transakciju nakon što su svi bajtovi pročitani.

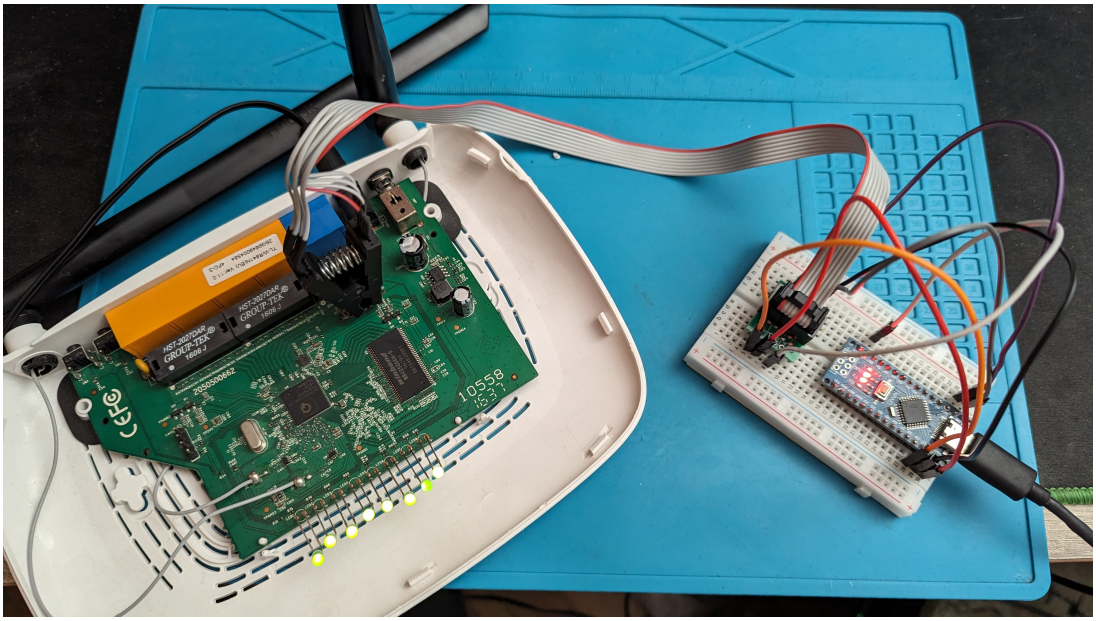
U funkciji readEEPROM obrađuje se glavni dio SPI protokola :

- digitalWrite(EEPROM_CS, LOW) : postavlja CS pin na LOW i pruža komunikaciju sa EEPROM-om;
- SPI.transfer(0x03) : šalje se 0x03 odnosno „Read Data“ instrukcija;
- SPI.transfer((address » 16) & 0xFF) : adresa se pomiče za 16 bitova na desno i postavlja masku za najsignifikantniji bajt;
- SPI.transfer((address » 8) & 0xFF) : adresa se pomiče za 8 bitova na desno i postavljamo masku za srednji bajt;
- SPI.transfer(address & 0xFF) : adresa se pomiče za 8 bitova na desno i postavlja masku za zadnji bajt;
- rdata = SPI.transfer(0xFF) : nakon što je poslana read instrukcija i adresa, šalje se „dummy“ bajt koji govori EEPROM-u da vrati bajt postavljen na adresi;
- digitalWrite(EEPROM_CS, HIGH) : postavlja CS na HIGH te onemogućuje EEPROM i komunikaciju;
- return rdata : vraća dohvaćeni bajt koji se ispisuje.

Bit shift se odvija jer EEPROM prima adresu u 3 odvojena bajta. Kako su u EEPROM-u 24-bitne adrese, na ovaj način se šalje cijela adresa (24b = 3B). Nakon napisane skripte, spajamo EEPROM s Arduinom prema dijagramu u tablici 2 , što se vidi na slici 14.

Zatim, spajamo Arduino na računalo i otvaramo Serial monitor na Visual Studio Code-u. Serial monitor omogućuje pohranu outputa arduina u txt obliku koji će se kasnije analizirati. Sljedeći koraci su uključenje uređaja i izvršavanje skrpe. Nakon završetka, preimenujemo txt dokument u firmware.bin za daljnju analizu. Otvaramo WSL s instaliranim Ubuntu sistemom te pokrećemo program binwalk, vidljiv na isječku 9 čiji rezultat vidimo na isječku 10.

Firmware se sastoji od različitih vrsta podataka, kao što su U-Boot version strin, CRC32 polinomsku tablicu, uImage zaglavlje (što govori da se koristi Linux kernel), LZMA komprimirane



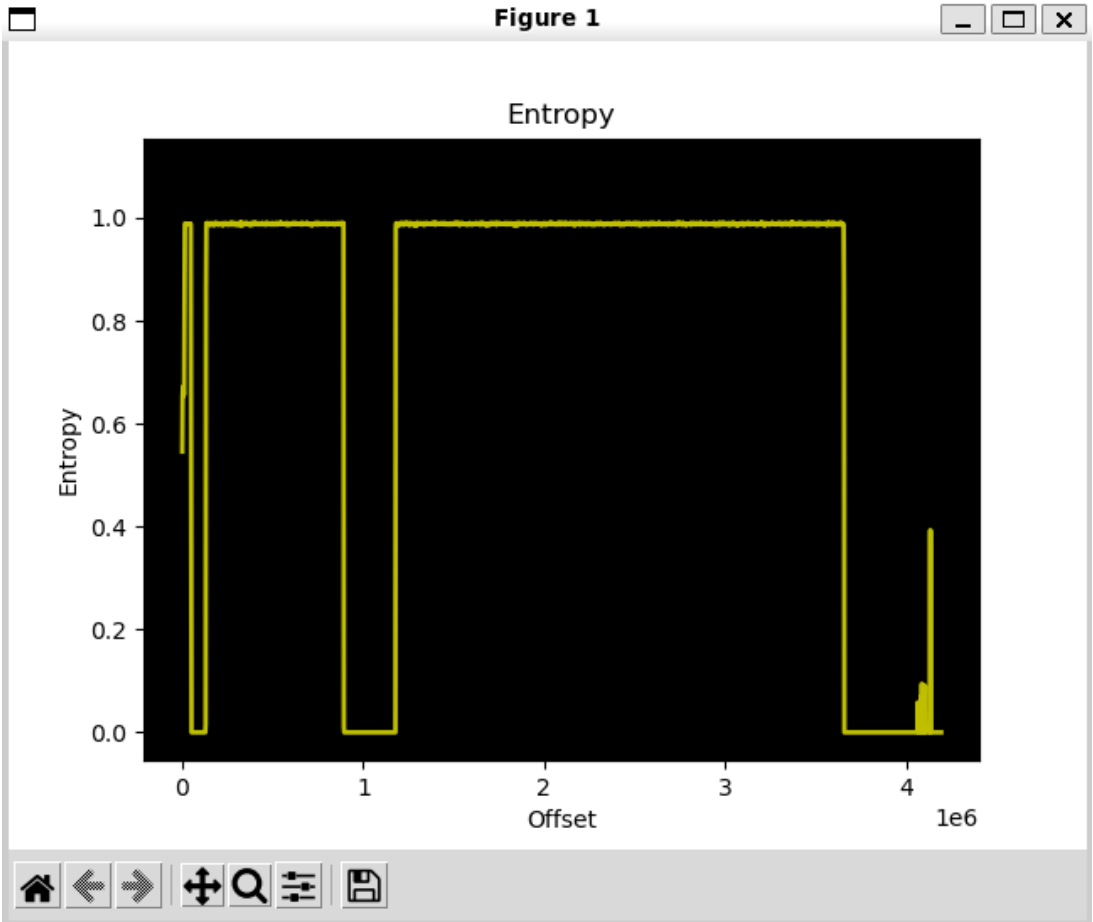
Slika 14: Spajanje EEPROM-a putem SPI na Arduino; autorska slika

```
binwalk firmware.bin
```

Isječak koda 9: Binwalk komanda

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
12928         0x3280          U-Boot version string, "U-Boot 1.1.4 (Mar 10 2015 - 15:00:39)"
12976         0x32B0          CRC32 polynomial table, big endian
14288         0x37D0          uImage header, header size: 64 bytes, header CRC: 0x8E2B46CA, created: 2015-03-10 07:00:39, image size: 35711 bytes, Data Address: 0x80010000, Entry Point: 0x80010000, data CRC: 0x72C78246, OS: Linux, CPU: MIPS, image type: Firmware Image, compression type: lzma, image name: "u-boot image"
14352         0x3810          LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 93256 bytes
131072        0x20000         TP-Link firmware header, firmware version: 0.0.3, image version: "", product ID: 0x0, product version: 138477584, kernel load address: 0x0, kernel entry point: 0x80002000, kernel offset: 3932160, kernel length: 512, rootfs offset: 761358, rootfs length: 1048576, bootloader offset: 2883584, bootloader length: 0
131584        0x20200         LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 2219160 bytes
1179648       0x120000        Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 2477651 bytes, 560 inodes, blocksize: 131072 bytes, created: 2015-03-10 07:25:11
```

Isječak koda 10: Rezultat binwalk-a



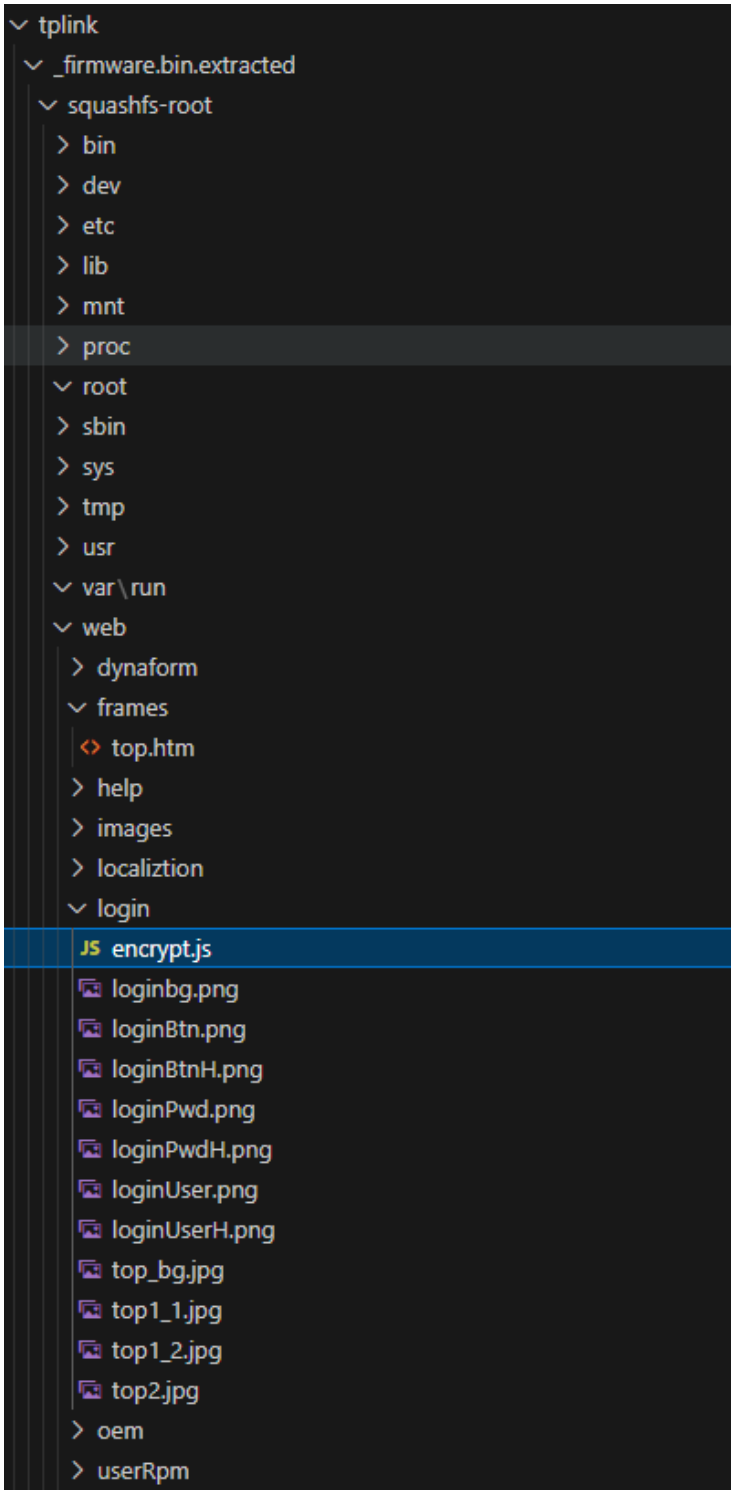
Slika 15: Entropija firmware.bin datoteke; autorska slika

podatke, TP-Link firmware i Squashfs datotečni sustav. Danjom analizom gledamo entropiju firmawre-a koristeći zastavicu „-E“ uz binwalk 15. Dijelovi s visokom entropijom (blizu 1) označavaju da su to komprimirani podaci. Dalje, stavljamo zastavicu „-e“ kako bi izvukli datotečni sustav. Pokretanjem binwalk -e firmware.bin dobiva se rezultat vidljiv (slika 16). Na slici se vidi file sistem koji se nalazi na routeru. Neke od zanimljivijih datoteka su ekrani admin-a prilikom konfiguracije, konfiguracijske postavke routera i javascrip datoteka s algoritmom enkripcije.

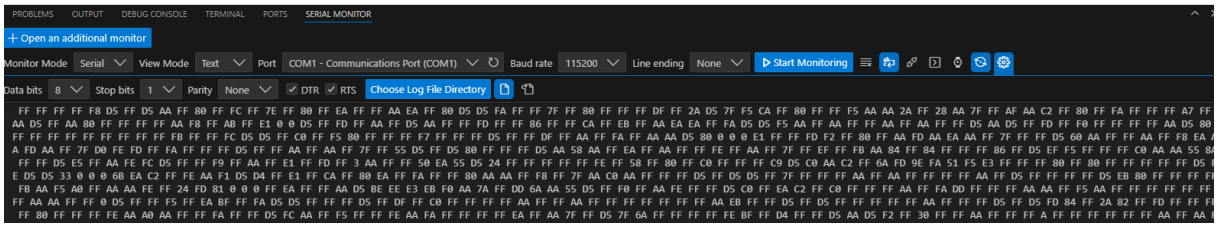
Uz podatke nađene online, jeftini hardware i malo programiranja dobivamo cijeli file sistem koji se može analizirati i reverse engineer-ati. Shodno tome, nije teško naći rupe u zaštiti koje bi mogle omogućiti probijanje zaštite remote.

5.2.2. DIGOO Wifi kamera

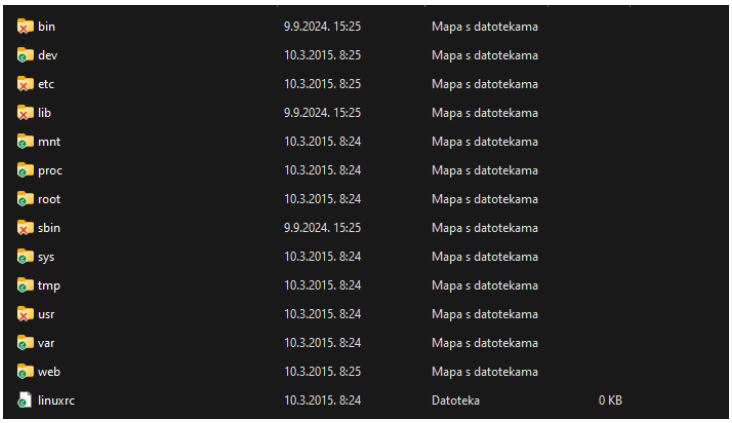
Sljedeći uređaj koji će biti analiziran je DIGOO Wifi kamera. Online potraga za poznate ranjivosti ne nudi puno članaka o uređaju, stoga je idući korak otvaranje uređaja. Uređaj se sastoji od 3 tiskane ploče. Srednja ploča sadržava ULN2803AG red tranzistora, xjc520fx0 IPC i 25Q64JVSIQ 8MB flash chip. Nakon istraživanja imena EEPROM-a nailazimo na dokumentaciju u kojoj je navedeno da EEPROM implementira SPI protokol u modu 0 i 3. Pinovi su drugog imena u dokumentaciji, ali se nalaze na istim pozicijama kao i prethodni router.



Slika 16: Rezultat ekstrakcije firmware.bin datoteke; autorska slika



Slika 17: Dohvaćanje firmware-a; autorska slika



bin	9.9.2024. 15:25	Mapa s datotekama	
dev	10.3.2015. 8:25	Mapa s datotekama	
etc	10.3.2015. 8:25	Mapa s datotekama	
lib	9.9.2024. 15:25	Mapa s datotekama	
mnt	10.3.2015. 8:24	Mapa s datotekama	
proc	10.3.2015. 8:24	Mapa s datotekama	
root	10.3.2015. 8:24	Mapa s datotekama	
sbin	9.9.2024. 15:25	Mapa s datotekama	
sys	10.3.2015. 8:24	Mapa s datotekama	
tmp	10.3.2015. 8:24	Mapa s datotekama	
usr	10.3.2015. 8:24	Mapa s datotekama	
var	10.3.2015. 8:24	Mapa s datotekama	
web	10.3.2015. 8:25	Mapa s datotekama	
linuxrc	10.3.2015. 8:24	Datoteka	0 KB

Slika 18: Datotečni sustav Digoo kamere; autorska slika

Modificiramo kod tako da se dodaje fukncija koja provjerava je li EEPROM zauzet u trenutku slanja instrukcija i mijenjamo for petlju na 8MB (8388608 B) vidljivo na isječku 11. Istom procedurom kao i prošli uređaj, arduino spajamo na računalo, otvaramo serial monitor na Visual Studio Code-u i odabiremo da želimo spremiti output. Pokrećemo dohvaćanje i spremamo datoteku kao firmware.bin. Serial monitor ispisuje stanje memorije vidljivo na slici 17. Nakon analize binwalk-a, uočavaju se čitljivi header-i te izvlačimo datotečni sustav vidljivo na slici 18.

Kako dosta uređaja implementira SPI na svojim EEPROM-ovima, uz male modifikacije na arduino kodu, dosta se jednostavno mogu dohvatiti firmware od različitih uređaja.

5.3. Debug pomoću SWD protokola

U ovom će poglavlju biti demonstrirani flash-anje i debug-iranje putem SWD (Serial wire debuging) protokola. Za potrebe ovog rada, flash-at ćemo software na ARM baziran Raspberry Pi Pico koristeći Raspberry Pi 3B+ kao host računalo. [21]

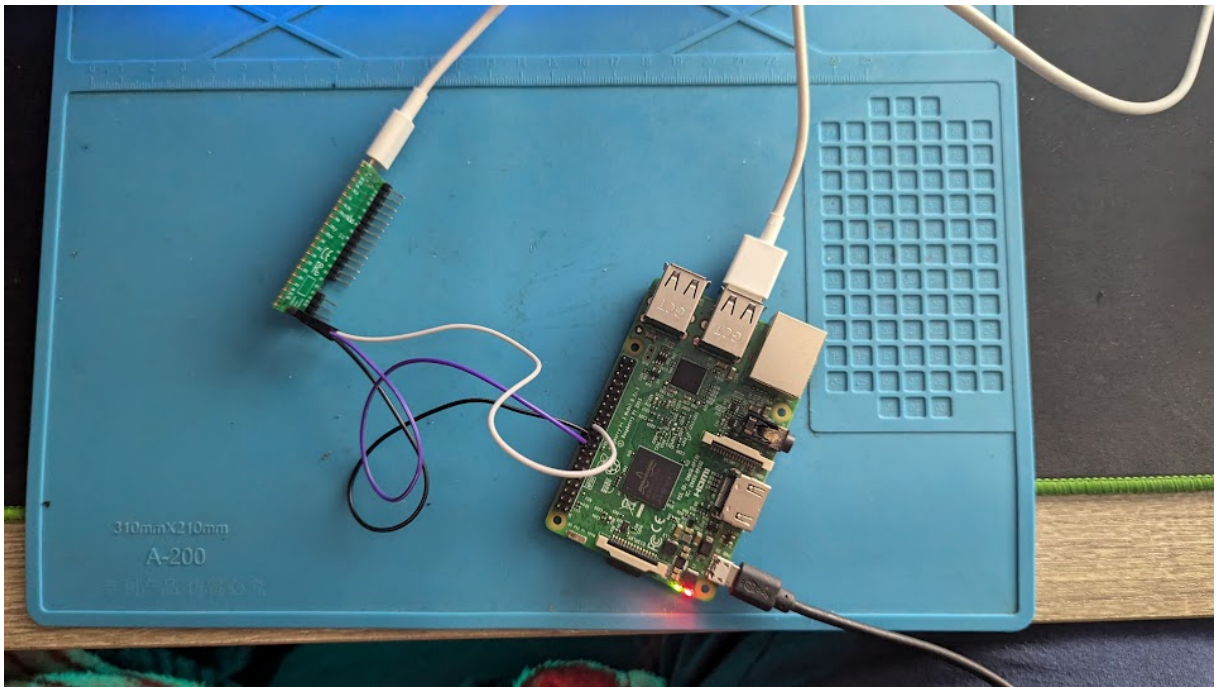
Za početak, pripremamo host računalo s potrebnim software-om za komunikaciju putem SWD-a. Prvo će se instalirati OpenOCD (On Chip Debugger [22]) tako da kloniramo repozitorij s https://github.com/raspberrypi/openocd.git iz branch-a RP2040 (što je mikrokontroler korišten na Raspberry Pi Pico). Dalje, instaliramo GDB Multiarch (GNU Debugger) kako bi mogli debug-irat software na chipu.

```
#include <SPI.h>
#define EEPROM_CS 10
#define READ_DATA 0x03
#define READ_STATUS_REG 0x05

void setup() {
  Serial.begin(115200);
  pinMode(EEPROM_CS, OUTPUT);
  SPI.begin();

  SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));

  for (long i = 0; i < 8388608; i++) {
    if (!isEEPROMBusy()) {
      byte data = readEEPROM(i);
      Serial.print(data, HEX);
      Serial.print(" ");
    }
  }

  SPI.endTransaction();
}

bool isEEPROMBusy() {
  digitalWrite(EEPROM_CS, LOW);
  SPI.transfer(READ_STATUS_REG);
  byte status = SPI.transfer(0xFF);
  digitalWrite(EEPROM_CS, HIGH);
  return (status & 0x01);
}

byte readEEPROM(long address) {
  byte rdata = 0xFF;
  digitalWrite(EEPROM_CS, LOW);
  SPI.transfer(READ_DATA);
  SPI.transfer((address >> 16) & 0xFF);
  SPI.transfer((address >> 8) & 0xFF);
  SPI.transfer(address & 0xFF);
  rdata = SPI.transfer(0xFF);
  digitalWrite(EEPROM_CS, HIGH);
  return rdata;
}

void loop() {}
```

Isječak koda 11: Arduino kod za SPI komunikaciju na 25Q64JVSIQ

RPi Pico	RPi 3
SWDIO	GPIO 24 (pin 18)
SWD GND	GND (pin 20)
SWCLK	GPIO 25 (pin 22)

Tablica 3: Dijagram spajanja SWD na Raspberry Pi



Slika 19: Spajanje RaspberryPi pico putem SWD; autorska slika

Spajamo Raspberry Pi Pico na Raspberry Pi 3 prema dijagramu 3 što je prikazano na slici 19. Nakon spajanja uređaja na host računalo, na host računalu šaljemo komandu 12.

Komanda radi sljedeće : *openocd* (poziva alat OpenOCD), *-f interface/raspberrypi-swd.cfg* (specificira konfiguracijsku datoteku za sučelje na hardware-u), *target/rp2040.cfg* (specificira konfiguracijsku datoteku za mikrokontroler; u ovom slučaju prije navedeni RP2040 chip), *-c "program blink/blink.elf verify reset exit"*. U zadnjem navedenoj komandi, "-c" zastavica omogućava komande koje će se izvršiti nakon što se file učita. Naime, prvo definiramo program za flash-anje "blink.elf". Dalje, verificiramo je li program dobro učitan i resetiramo mikrorontroler). Na kraju završavamo s komandom "exit". Rezultat je vidljiv u isječku 13. Pošto na uređaju treperi LED, zaključak je da je program uspješno flash-an.

U nastavku ćemo proći debug-iranje putem SWD-a. Prije pokretanja GDB servera, Pico se spaja putem UART-a na host mašinu. Razlog tome je to što USB neće raditi kada

```
openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg -c
↪  "program blink/blink.elf verify reset exit"
```

Isječak koda 12: OpenOCD komanda za flashanje blink.elf programa

```
Open On-Chip Debugger 0.11.0-g8e3c38f (2024-09-15-10:54)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
adapter speed: 1000 kHz

Info : Hardware thread awareness created
Info : Hardware thread awareness created
Info : RP2040 Flash Bank Command
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : clock speed 1001 kHz
Info : SWD DPIDR 0x0bc12477
Info : SWD DLPIDR 0x00000001
Info : SWD DPIDR 0x0bc12477
Info : SWD DLPIDR 0x10000001
Info : rp2040.core0: hardware has 4 breakpoints, 2 watchpoints
Info : rp2040.core1: hardware has 4 breakpoints, 2 watchpoints
Info : starting gdb server for rp2040.core0 on 3333
Info : Listening on port 3333 for gdb connections
target halted due to debug-request, current mode: Thread
xPSR: 0xf1000000 pc: 0x000000ea msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0xf1000000 pc: 0x000000ea msp: 0x20041f00
** Programming Started **
Info : RP2040 B0 Flash Probe: 2097152 bytes @10000000, in 512 sectors

target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
Info : Writing 8192 bytes starting at 0x0
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
** Programming Finished **
** Verify Started **
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00000184 msp: 0x20041f00
** Verified OK **
** Resetting Target **
shutdown command invoked
```

Isječak koda 13: OpenOCD log nakon flash-anja

RPi Pico	RPi 3
GPIO 0 (TX)	GPIO 15 (RX, pin 10)
GPIO 1 (RX)	GPIO 14 (TX, pin 8)
GND	GND (pin 14)

Tablica 4: Dijagram spajanja UART na Raspberry Pi

```
openocd -f interface/raspberrypi-swd.cfg -f target/rp2040.cfg
```

Isječak koda 14: OpenOCD pokretanje GBD servera

je procesor u debug načinu koristeći dijagram 4. Za početak, pokrećemo OpenOCD GDB server na koji će se spajati klijenti za potrebe debugiranja. Koristimo komandu 14. Nadalje, otvaramo novu konekciju na host računalo i pokrećemo GDB klijent komandom 15. Nakon što je klijent otvoren, u GDB terminalu šaljemo target, odnosno server gdje je poslužen GDB server. Loadamo program koristeći komande target i load 16. Na ekranu 20 se vidi uspješno uspostavljena konekcija na kojoj se mogu davati standardne GDB komande kako bismo istražili rad mikrokontrolera.

5.4. Firmware dump pomoću JTAG protokola

Posljednji protokol koji će biti odrađen je JTAG (Joint Test Action Group) protokol. Uređaj koji će biti analiziran je Linksys WRT54G v2.0 router. U fazi prikupljanja informacija, pronađeno je da uređaj ima Broadcom BCM4712 procesor i 4 MiB Intel TE28F320C3BD70 flash memorije. Također, pronađena je slika matične ploče koja otkriva dvije grupe izloženih kontaktnih površina označene sa JP1 i JP2. Istraživanjem je nađena informacija da JP1 sadrži kontakte za UART konekciju (Tx i Rx pinove). Zatim, JP2 grupa je JTAG konekcija vjerojatno ostavljena prilikom proizvodnje uređaja za verifikaciju ispravnosti. U sljedećoj fazi, otvaramo uređaj prilikom čega pronalazimo te kontaktne površine. JP1 Tx i Rx pinovi se spajaju na adapter te čitamo output na terminalu. Sada se mogu vidjeti osnovne informacije o bootloaderu, mapiranje memorije i najbitnije, početnu adresu flash memorije 17.

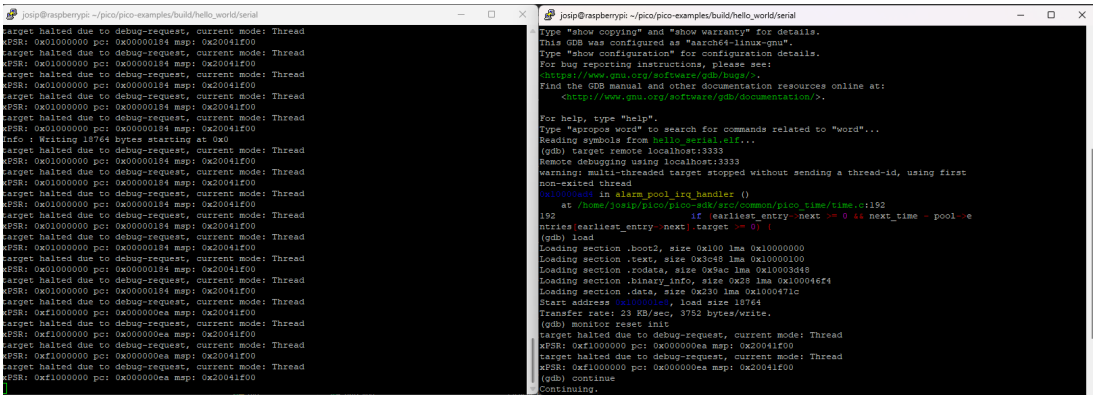
Za kreiranje JTAG konekcije, potrebni su software i hardware. Tokom istraživanja, odabran je UrJTAG [23] koji u svojoj listi target chipova sadrži BCM4712, chip na uređaju. Za hardware-e je odabran Raspberry Pi Pico s flashanim DirtyJTAG projektom [24]. Razlog odabira je bio prikazati da je moguće ostvariti željeni rezultat s jeftinim hardware-om. Pinovi s Raspberry Pi Pico se spajaju prema sljedećem dijagramu 5. Kako to izgleda u praksi, prikazano je na slici 21.

```
gdb-multiarch hello_serial.elf
```

Isječak koda 15: GDB multiarch spajanje klijenta

```
(gdb) target remote localhost:3333
(gdb) load
```

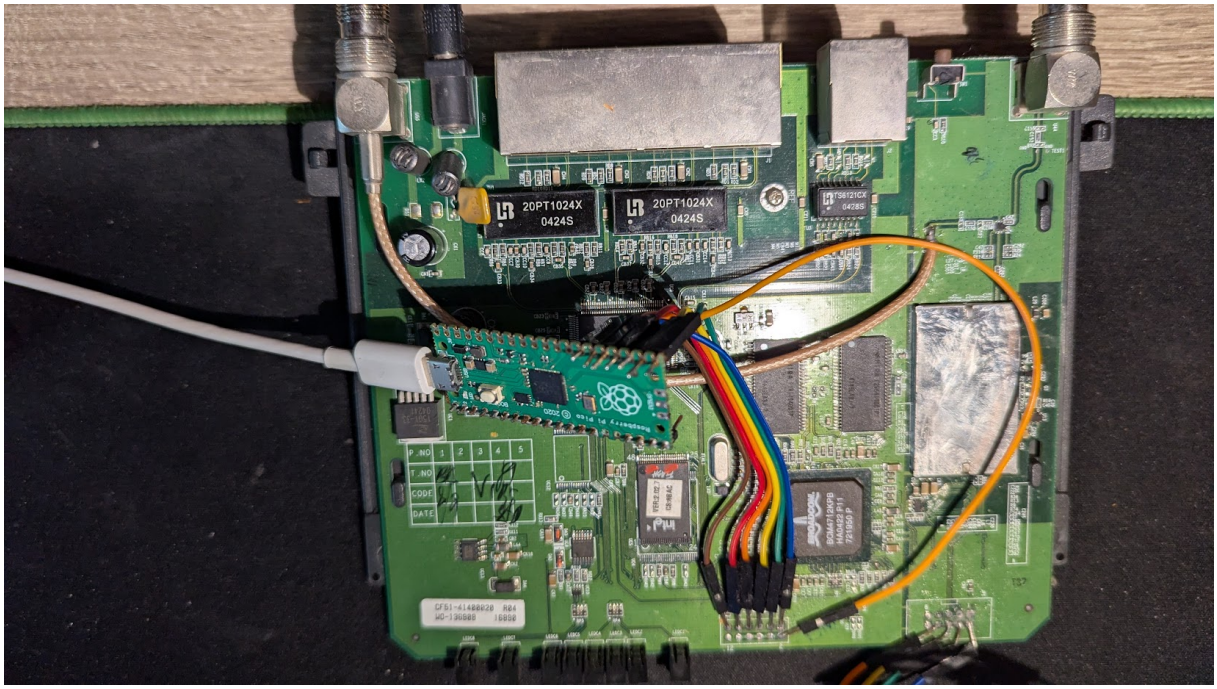
Isječak koda 16: GDB multiarch spajanje na server



Slika 20: Debug putem SWD-a; autorska slika

JP2 pin	RPi Pico
1 - nTRST	GP21
3 - TDI	GP16
5 - TDO	GP17
7 - TMS	GP19
9 - TCK	GP18
11 - nSRST	GP20
2 - GND	GND

Tablica 5: Dijagram spajanja JTAG na Raspberry Pi Pico



Slika 21: Konekcija JTAG između WRT54G routera i Raspberry Pi Pico; autorska slika

```
Initializing Arena.
Initializing Devices.
et0: Broadcom BCM47xx 10/100 Mbps Ethernet Controller 3.50.21.0
CPU type 0x29007: 200MHz
Total memory: 0x2000000 bytes (32MB)

Total memory used by CFE:  0x80334DC0 - 0x8043A310 (1070416)
Initialized Data:          0x80334DC0 - 0x80336F40 (8576)
BSS Area:                  0x80336F40 - 0x80338310 (5072)
Local Heap:                0x80338310 - 0x80438310 (1048576)
Stack Area:                0x80438310 - 0x8043A310 (8192)
Text (code) segment:       0x80300000 - 0x8030F220 (61984)
Boot area (physical):      0x0043B000 - 0x0047B000
Relocation Factor:         I:00000000 - D:00000000

Boot version: v2.3
The boot is CFE

mac_init(): Find mac [00:12:17:07:68:01] in location 1
Nothing...
Device eth0:  hwaddr 00-12-17-07-68-01, ipaddr 192.168.1.1, mask
 ↪  255.255.255.0
        gateway not set, nameserver not set
Reading :: Failed.: Timeout occured
Loader:raw Filesys:raw Dev:flash0.os File: Options:(null)
Loading: .. 3856 bytes read
Entry at 0x80001000
Closing network.
Starting program at 0x80001000
CPU ProcId is: 0x00029007, options: 0x0000004d

...

1: offset=0x10000,size=0x10000,blocks=63

Using word write method

Flash device: 0x400000 at 01c000000
```

Isječak koda 17: UART output za WRT54G router

Prije pokretanja UrJTAG software moramo proslijediti USB uređaj na WSL. Za to se koristi *usbipd*. Prvo ispisujemo uređaje pomoću *usbipd list* gdje tražimo DirtJtag uređaj i njegov BUSID. Dalje, spajamo uređaj i dodjeljujemo ga virtualnoj mašini (WSL) pomoću *usbipd bind –busid <BUSID>* i *usbipd attatch –wsl –busid <BUSID>*. Kako bi se utvrdilo da je uređaj prepoznatljiv na WSL-u, koristimo *lsubs*. Zaključak je da se nalazi u listi uređaja.

Idući koraci su uključivanje router-a, spojanje Raspberry Pi Pico na računalo i pokretanje UrJTAG software-a. Zatim, prvo je potrebno spojiti uređaj za komunikaciju pomoću *cable dirtyjtag*. Provjera konekcije se izvodi putem *detect* naredbe gdje potvrđujemo da RPi Pico vidi targetirani procesor. Pošto je targetirani procesor MIPS arhitekture, možemo inicijalizirati ejtag (Enhanced JTAG) konekciju. Ona omogućuje high-level naredbe nad procesorom pomoću *initbus ejtag*. Pošto je procesor u debug modu, pokrećemo naredbu *readmem 0x1c000000 0x400000 firmware.bin*. Naredba čita sadržaj memorije počevši od adrese 0x1c000000, koja je nađena čitajući UART output, i njezinu duljinu 4MB što je veličina flash memorije. Također, pohranjuje rezultat datoteku firmware.bin. Output se vidi na isječku 18. Nakon preuzimanja sadržaja firmware-a, radimo već objašnjenu analizu software-a pomoću binwalk-a. Dobiven je file sistem, što nam je bio cilj.

5.5. Načini zaštite

5.5.1. Zaštita od UARTa

Prvi korak u zaštiti je onemogućavanje UART-a u produkcijskim verzijama uređaja ako nije potreban. No, mnogi proizvođači ostavljaju aktivne UART konektore za potrebe razvoja i testiranja. Ako se ne deaktiviraju prije isporuke, predstavljaju ozbiljan sigurnosni rizik. Kada se UART mora koristiti u produkciji, preporučuje se autentifikacija korisnika prilikom pristupa serijskoj konzoli, čime se sprječava neovlašteni ulaz.

Nadalje, korisnike se može zaštiti na razne načine. Implementacija sigurnog boot mehanizma osigurava da samo ovjereni firmware može biti učitan. Time se sprječava manipulacija sustavom čak i ako napadač uspije dobiti pristup putem UART-a. Također, šifriranje serijske komunikacije pomoću sigurnih protokola ili hardwareske enkripcije može poboljšati zaštitu od neželjenih napada. Kako bi se otežalo identificiranje UART pinova, tiskana pločica (eng. printed circuit board, PCB) se može fizički zaštiti. Primjer toga je uklanjanje neovlaštenih kontaktnih površina ili njihovo maskiranje [25]. Kombinacija ovih zaštitnih mjera značajno smanjuje sigurnosni rizik i osigurava da ugrađeni sustavi ostanu otporni na napade koji koriste ovaj serijski protokol.

5.5.2. Zaštita od SPI

Uređaji često koriste SPI Flash memoriju za pohranu firmware-a. Zbog toga, napadači mogu koristiti SPI interface za dohvaćanje i modificiranje firmware-a ako su priključne točke dostupne na PCB-u. Jedan od najvažnijih koraka u zaštiti SPI komunikacije je spriječiti izravan pristup SPI memoriji kada to nije nužno.

```
jtag> cable dirtyjtag
jtag> detect
IR length: 8
Chain length: 1
Device Id: 00010100011100010010000101111111 (0x000000001471217F)
    Manufacturer: Broadcom
    Part(0): BCM4712
    Stepping: Ver 1
    Filename: /opt/local/share/urjtag/broadcom/bcm4712/bcm4712
jtag> initbus ejtag
ImpCode=00000000100000000000100100000100 00800904
EJTAG version: <= 2.0
EJTAG Implementation flags: R4k DMA MIPS32
Processor entered Debug Mode.
jtag> detectflash 0x1c000000
Query identification string:
Primary Algorithm Command Set and Control Interface ID Code: 0x0003
 ↪  (Intel Standard Command Set)
Alternate Algorithm Command Set and Control Interface ID Code: 0x0000
 ↪  (null)
Query system interface information:
Vcc Logic Supply Minimum Write/Erase or Write voltage: 2700 mV
Vcc Logic Supply Maximum Write/Erase or Write voltage: 3600 mV
Vpp [Programming] Supply Minimum Write/Erase voltage: 11400 mV
Vpp [Programming] Supply Maximum Write/Erase voltage: 12600 mV
Typical timeout per single byte/word program: 32 us
Typical timeout for maximum-size multi-byte program: 0 us
Typical timeout per individual block erase: 1024 ms
Typical timeout for full chip erase: 0 ms
Maximum timeout for byte/word program: 512 us
Maximum timeout for multi-byte program: 0 us
Maximum timeout per individual block erase: 8192 ms
Maximum timeout for chip erase: 0 ms
Device geometry definition:
Device Size: 4194304 B (4096 KiB, 4 MiB)
Flash Device Interface Code description: 0x0001 (x16)
Maximum number of bytes in multi-byte program: 1
Number of Erase Block Regions within device: 2
Erase Block Region Information:
    Region 0:
        Erase Block Size: 8192 B (8 KiB)
        Number of Erase Blocks: 8
    Region 1:
        Erase Block Size: 65536 B (64 KiB)
    Number of Erase Blocks: 63
jtag> readmem 0x1c000000 0x400000 firmware.bin
address: 0x1C000000
length: 0x00400000
reading:
addr: 0x3C20F000
```

Isječak koda 18: UrJTAG konekcija

Kontrola pristupa kroz softwareske sigurnosne mehanizme se izvodi na mnoge načine. Na primjer, autentifikacija prilikom pokretanja komunikacije između SPI master-a i slave-a smanjuje rizik neovlaštenog pristupa. Nadaje, nedozvoljene promjene podataka se mogu spriječiti zaključavanjem ili zaštitom od zapisivanja (eng. write protection) pomoću sigurnosnih bitova na SPI memoriji. Osim toga, implementacija enkripcije podataka prije prijenosa putem SPI veze značajno povećava sigurnost. U ovom slučaju, čak i ako napadač uspije presresti komunikaciju, podatci će biti nečitljivi bez odgovarajućeg ključa. Dodatan način zaštite je provjera integriteta firmware-a, kao što je Secure Boot. Ono osigurava da sustav može učitati samo ovjereni i sigurni firmware, sprječavajući napade koji uključuju zamjenu ili modifikaciju datoteka u SPI memoriji. [26]

Zaštita na hardwareskoj razini također igra važnu ulogu. Na primjer, fizičko uklanjanje ili skrivanje SPI kontaktnih površina na PCB-u može spriječiti izravno spajanje napadača na komunikacijske linije.

5.5.3. Zaštita od JTAG i SWD

Najefikasnija zaštita je potpuno onemogućavanje JTAG/SWD protokola u konačnoj produkcijskoj verziji uređaja. To se može postići fizičkim uklanjanjem JTAG/SWD pinova s PCB-a. Također, većina modernih mikrokontrolera i procesora omogućuje JTAG/SWD Lock Bit opciju, koja trajno deaktivira JTAG/SWD nakon završetka razvoja.

U slučaju kada JTAG/SWD mora ostati aktivan za potrebe održavanja, preporuča se zaštita pristupa putem lozinke ili autentifikacije. Time se regulira neovlašteni pristup, tj. samo ovlašteni korisnici mogu koristiti debug funkcionalnosti. Dodatna sigurnosna mjera je zaključavanje JTAG-a/SWD-a pomoću softwareskih ili hardwareskih sigurnosnih mehanizama. Za napredniju zaštitu se može koristiti obfuskacija hardwareskog dizajna, što otežava analizu i inverzni inženjering JTAG/SWD sučelja. Fizički sigurnosni mehanizmi, poput otkrivanja pokušaja manipulacije uređajem, mogu automatski deaktivirati JTAG/SWD ili čak izbrisati osjetljive podatke ako se detektira neovlašteni pristup. Nadalje, Secure Boot učitava samo verificirani firmware. Čak i ako napadač dobije pristup JTAG/SWD sučelju, Secure Boot neće moći učitati neovlašteni firmware. [27] Fizička zaštita je također ključna. Uklanjanje ili skrivanje JTAG/SWD kontaktnih površina na PCB-u može otežati njihovo otkrivanje i spajanje napadača na debug port.

6. Zaključak

Sigurnost ugrađenih sustava postaje sve značajnije područje istraživanja i primjene, s obzirom na porast broja povezanih uređaja i rastuću složenost kibernetičkih prijetnji. U ovom su diplomskom radu analizirani ključni aspekti hardwareske sigurnosti, uključujući ranjivosti, napade i obrambene metodologije. Posebna pozornost je posvećena praktičnim aspektima analize sigurnosti ugrađenih sustava, koristeći alate i tehnike penetracijskog testiranja i analize firmware-a.

Provedena istraživanja i eksperimenti pokazali su da su mnogi ugrađeni sustavi podložni napadima zbog nedovoljno implementiranih sigurnosnih mjera. Demonstrirano je kako neovlašteni pristup putem sučelja, poput UART-a, JTAG-a i SPI-ja, može omogućiti preuzimanje i modifikaciju firmware-a, čime se otvara prostor za potencijalne zlouporabe. Integracija sigurnosnih mjera već u fazi dizajna hardware-a je ključna za smanjenje rizika od fkompromitacije.

Zaključno, rezultati ovog rada potvrđuju potrebu za kontinuiranim razvojem i unapređenjem hardwareske sigurnosti. S obzirom na brzo rastući broj IoT uređaja i njihovu sve veću primjenu u industriji, zdravstvu i kritičnoj infrastrukturi, sigurnosni propusti u hardware-u mogu imati ozbiljne posljedice. Buduća istraživanja trebala bi se usmjeriti na daljnji razvoj metoda detekcije hardwareskih trojanaca, unapređenje tehnika mitigacije bočnih napada i razvoj sigurnijih arhitektura ugrađenih sustava.

Kombinacija akademskog istraživanja i praktičnih testiranja ključna je za postizanje sigurnih i pouzdanih hardwareskih rješenja u budućnosti.

Popis literature

[1] N. Dhanjani, *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts*. O'Reilly Media, Incorporated, 2015., ISBN: 9781491902332.

[2] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2020., ISBN: 9781119642787.

[3] M. Wolf i T. Gendrullis, „Designing Secure Hardware," *IEEE Design & Test of Computers*, 2011.

[4] Hackster.io, *ESP32 Plant Monitoring with Arduino IoT Cloud & Remote App*, Accessed: 2024-07-15, 2024.

[5] Kaspersky, *Best Practices for IoT Security*, Accessed: 2024-07-16, 2024.

[6] J.-G. Valle, *Practical Hardware Pentesting: A guide to attacking embedded systems and protecting them against the most common hardware attacks*. Packt Publishing Ltd, 2021.

[7] J. Van Woudenberg i C. O'Flynn, *The Hardware Hacking Handbook: Breaking Embedded Security with Hardware Attacks*. No Starch Press, 2021.

[8] K. Yang, Y. Hu, D. Chen i dr., „Machine Learning-Enhanced Analysis and Design for Trustworthy Integrated Circuits," *AI-Enabled Electronic Circuit and System Design: From Ideation to Utilization*, Springer, 2025., str. 467–496.

[9] Microsoft, *Why Visual Studio Code?* Accessed: 2024-09-20, 2024.

[10] PuTTY Team, *PuTTY: A free SSH and Telnet client*, Accessed: 2024-10-01, 2024.

[11] Microsoft, *Windows Subsystem for Linux FAQ*, Accessed: 2024-10-01, 2024.

[12] Arduino, *Arduino IDE v1 Basics*, Accessed: 2024-10-15, 2024.

[13] Intel Community, *About OpenOCD*, Accessed: 2024-10-15, 2024.

[14] Arduino, *Arduino Uno Rev3*, Accessed: 2024-10-15, 2024.

[15] Raspberry Pi Foundation, *Raspberry Pi Pico*, Accessed: 2024-11-02, 2024.

[16] Raspberry Pi Foundation, *Raspberry Pi 3 Model B*, Accessed: 2025-01-15, 2024.

[17] Components 101, *FT232RL USB to TTL Converter - Pinout, Features, Datasheet, Working & Application Alternative*, Accessed: 2024-11-02, 2024.

[18] Silicon Labs Community, *Serial Wire Debug (SWD)*, Accessed: 2025-01-15, 2024.

[19] F. C. C. (FCC), *FCC Report: FCC ID V7TF3*, Accessed: 2024-08-05, 2024.

[20] GigaDevice, *GD25Q32C Serial NOR Flash Memory*, Accessed: 2024-09-15, 2020.

[21] E. Hub, *Programming Raspberry Pi Pico with SWD (Serial Wire Debug)*, Accessed: 2024-09-16, 2024.

[22] OpenOCD, *Open On-Chip Debugger (OpenOCD)*, Accessed: 2024-09-15, 2024.

[23] U. D. Team, *UrJTAG: Universal JTAG Library and Tools*, Accessed: 2024-09-16, 2024.

[24] P. Dussud, *pico-dirtyJtag*, Accessed: 2024-09-20, 2024.

[25] Embedded, *How to Secure UART Communications in IoT Devices*, Accessed: 2025-02-13, 2024.

[26] Payatu, *Hardware Attack Surface: SPI*, Accessed: 2025-02-13, 2024.

[27] Asset Intertech, *Securing the JTAG Interface*, Accessed: 2025-02-13, 2019.

Popis slika

Popis tablica

Popis isječaka koda