

Elektrotehnički fakultet

Univerzitet u Banjoj Luci

**IZVJEŠTAJ PROJEKTNOG ZADATKA**

iz predmeta

**POUZDANOST ELEKTRONSKIH SISTEMA**

Student: Vidić Luka 2201/24

Mentor: Prof. Dr Ivanović Željko

# Sadržaj

[Sadržaj 2](#_Toc191155272)

[1. Uvod 3](#_Toc191155273)

[2. Ciljna platforma i hardverska struktura sistema 4](#_Toc191155274)

[2.1. NUC980 serija mikroprocesora 4](#_Toc191155275)

[3. Priprema operativnog sistema za ciljnu platformu 9](#_Toc191155276)

[3.1. *Buildroot* 9](#_Toc191155277)

[3.2. Konfiguracija Linux jezgra, Linux rukovaoci uređaja, struktura stabla uređaja 11](#_Toc191155278)

[3.2.1 Rukovaoci uređaja (*Linux Device Drivers*) 11](#_Toc191155279)

[3.2.2 Stablo uređaja (*Device Tree*) 12](#_Toc191155280)

[Literatura 14](#_Toc191155281)

# 1. Uvod

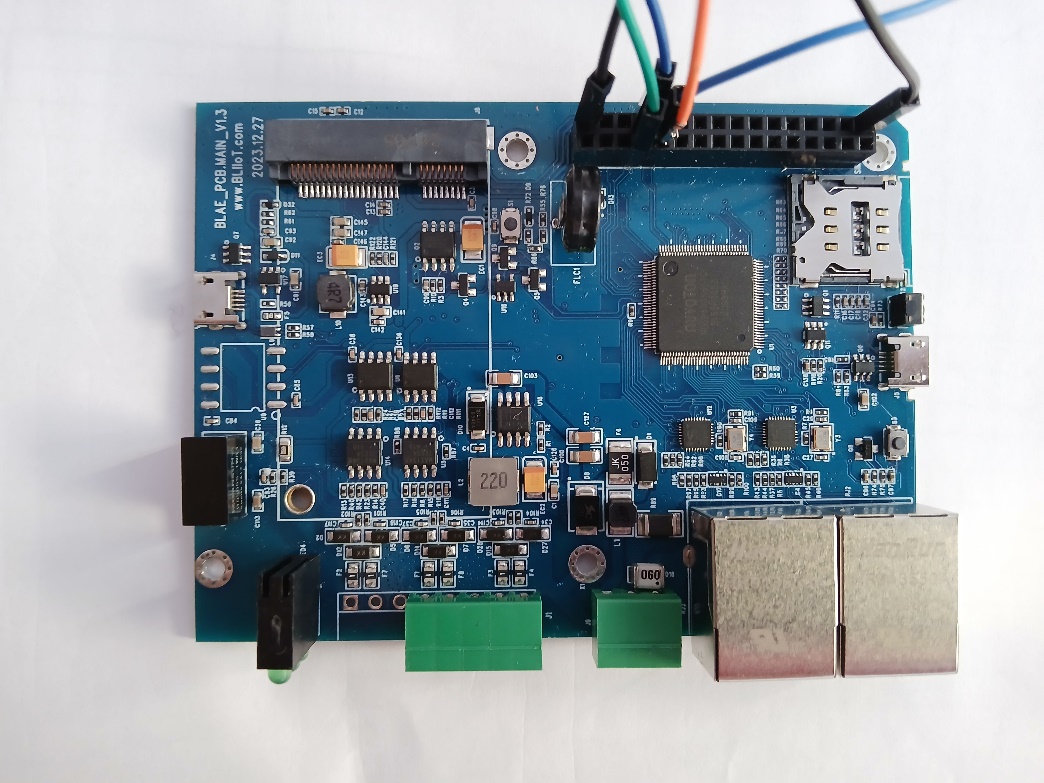
Cilj ovog projektnog zadatka je implementacija *gateway* uređaja između industrijskog protokola IEC 60870-5-104 (nadalje u dokumentu IEC 104) i industrijskog protokola MODBUS RTU. Uređaj je implementiran na komercijalno dostupnoj ploči koja je bazirana na NUC980DK61YC mikroprocesoru proizvođača Nuvoton. Svrha uređaja jeste mapiranje podataka, komandi i poruka IEC 104 protokola na protokol MODBUS RTU tako da se sa strane IEC 104 *master* uređaja (npr. računar sa SCADA sistemom) omogući razmjena podataka i slanje komandi sa MODBUS *slave* uređajima. Sam uređaj spada u široku grupu ugrađenih računarskih sistema te se implementacija sastoji od svih standardnih koraka potrebnih za implementaciju bilo koje aplikacije na ugrađenom sistemu:

* **Izučavanje hardvera** – Prvi i osnovni korak jeste upoznavanje sa hardverom sistema sa kojim se radi. U ovom slučaju razvojna ploča je komercijalna stoga kompletna dokumentacija nije javno dostupna već se izučavanje svodi na izučavanje pojedinačnih komponenti od značaja kao i odgovarajućih postupaka reverznog inženjeringa.
* **Priprema operativnog sistema na razvojnoj ploči** – Razvoj aplikacija na ugrađenim računarskim sistemima uglavnom se svodi na razvoj za ciljni operativni sistem koji će se izvršavati na razvojnoj ploči čime se omogućava portabilnost aplikacija na različite razvojne ploče. Najčešće korišćeni su Linux bazirani operativni sistemi ili neki jednostavniji RTOS operativni sistemi. U nekim slučajevima ovaj korak se može preskočiti, kada je u pitanju razvoj *bare-metal* aplikacija. U svrhu izrade ovog projekta korišćen je Linux operativni sistem.
* **Implementacija aplikacije** – Kada se okruženje pripremi za rad nastupa se sa izradom aplikacije. Izbor postojećih biblioteka, implementacija traženih funkcionalnosti te verifikacija istih su osnovni koraci ka implementaciji zahtjevane aplikacije. Za implementaciju ovog projekta glavne korištene biblioteke su *libiec60870* i *libmodbus*.

Kroz nastavak dokumenta biće detaljno opisani svi prethodno navedeni koraci i pojedinačni postupci sprovedeni u datim koracima.

# 2. Ciljna platforma i hardverska struktura sistema

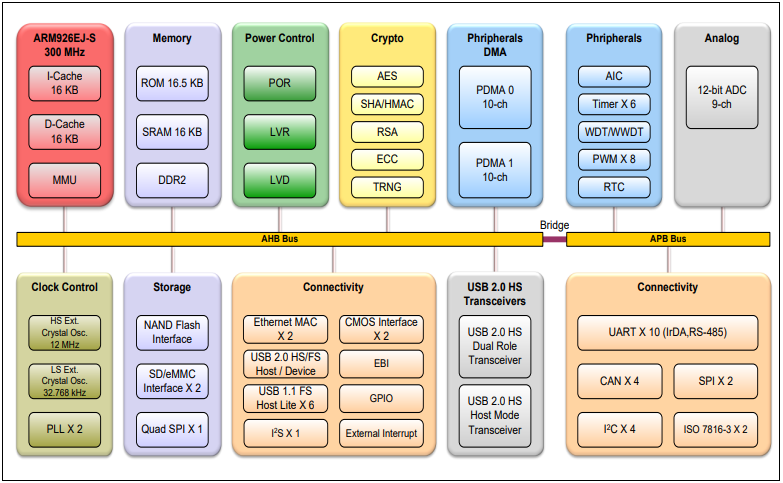
Hardverska implementacija sistema se zasniva na već izgrađenoj i komercijalno dospunoj štampanoj ploči koja je izvađena iz kućišta postojećeg uređaja (*gateway* slične namjene kao i u ovom projektu) proizvođača BLIIoT [1]. Data ploča prikazana je na slici 2.1.

  
Slika 2.1 – *Štampana ploča ciljne platforme*

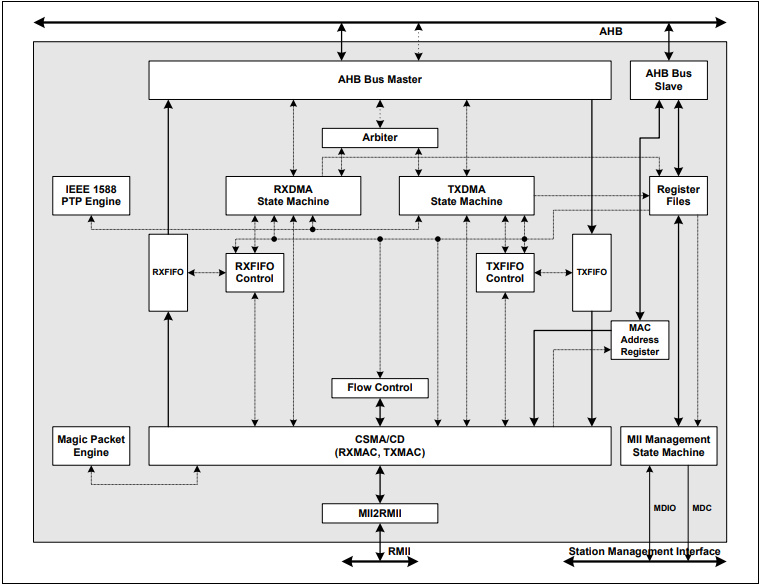
Data platforma bazirana je na mikroprocesoru NUC980DK61YC proizvođača Nuvoton koji predstavlja glavnu upravljačku jedinicu koja izvršava aplikativni program i upravlja periferijama od značaja. Platforma dolazi sa X MB fleš (*flash*) memorije X proizvođača *Winbond*, dva *Ethernet* porta koji su neophodni za IEC 104 komunikaciju te šest serijskih RS-485 kanala koji se koriste u svrhu MODBUS RTU komunikacije. Pored ovih glavnih karakteristika, ciljna platforma sadrži i dodatne kao što su interfejs za SIM karticu, dodatni serijski USB port u svrhe debagovanja, tastere za resetovanje i više LED za signalizaciju. U nastavku biće detaljnije prezentovana serija mikroprocesora NUC980 s obzirom na centralnu ulogu jednog takvog mikroprocesora u ovom sistemu.

## 2.1. NUC980 serija mikroprocesora

NUC980 serija mikroprocesora zasnovana je na ARM926EJ-S jezgru. Postoji nekoliko mikroprocesora iz date serije koji se razlikuju po broju pinova (pakovanju) i veličini interne DDR2 memorije. Mikroprocesori iz ove serije posebno su opremljeni velikim brojem Ethernet, UART, CAN i sličnih periferija koji ih čine dobrim izborom za industrijsku i IoT primjenu [2]. Na slici 2 prikazana je blok šema strukture NUC980 mikroprocesora koja daje detaljniji prikaz resursa i mogućnosti koje posjeduju mikroprocesori iz date serije.

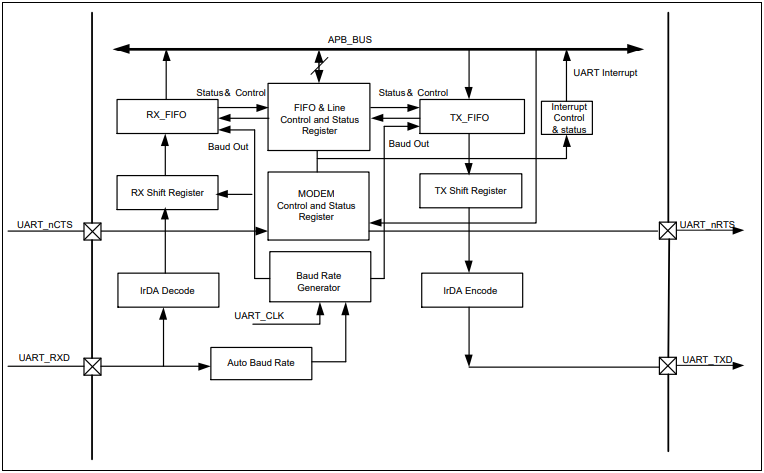
  
Slika 2.2 – *Blok šema mikroprocesora iz NUC980 serije* [3]

Za izradu ovog projektnog zadatka od najvećeg interesa je hardverska podrška za Ethernet (fizički sloj IEC 104 prokola zasnovan je na Ethernet-u) i podrška za UART, odnosno RS-485 (fizički sloj MODBUS RTU protokola zasniva se na nekom od serijskih protokola, u ovom slučaju RS-485). Što se tiče podrške za Ethernet, mikroprocesori iz NUC980 serije imaju dva Ethernet MAC kontrolera čime je hardverski podržana Ethernet komunikacija. Na slici 2.3 prikazana je struktura Ethernet kontrolera mikroprocesora iz NUC980 serije.

  
Slika 2.3 – *Ethernet MAC kontroler NUC980 serije mikroprocesora* [3]

Prvi Ethernet MAC kontroler zauzima prvih 10 pinova iz “E” banke pinova (E0 – E9), a drugi kontroler prvih 10 pinova iz “F” banke pinova (F0 – F9) [3]. Poznavanje ovog pinout-a je od značaja u sledećem koraku (priprema Linux operativnog sistema) kada se konfigurše struktura stabla uređaja (*device tree*). Ethernet portovi na ciljnoj ploči povezani su preko odgovarajućeg hardvera na ove pinove, čime je obezbjeđen interfejs za Ethernet komunikaciju (u slučaju datih portova brzina komunikacije je 100 Mbps).

NUC980 serija mikroprocesora obezbjeđuje čak deset kanala za UART komunikaciju pri čemu je hardverski podržan veliki broj konfiguracija i režima rada, između ostalog i željeni RS-485 režim. Blok šema UART kontrolera NUC980 serije mikroprocesora data je na slici 2.4.

  
Slika 2.4 – *UART kontroler NUC980 serije mikroprocesora* [3]

Za razliku od Ethernet kontrolera čiji su korišteni pinovi mikroprocesora jasno definisani, kod UART kontrolera RX i TX linije svakog kanala mogu da se konfigurišu tako da se podese različiti korišteni pinovi, još poznatije kao remapiranje pinova (*pin mapping*). Na primjer, kanal UART1 može za RX liniju da koristi pinove F9, A0 ili C6, dok kanal TX može da koristi F10, A1 ili C5. Dostupni pinovi za ostale kanale i linije jasno su navedeni u dokumentaciji proizvođača [3]. Što se tiče ciljne platforme, ono što je izazov jeste činjenica da je ona komercijalni proizvod te da dokumentacija, u kojoj je između ostalog i električna šema, nije javno dostupna. Da bi se pronašlo UART kanali se koriste kao i koji pinovi primjenjen je postupak reverznog inženjeringa, koji je podrazumijevao detekciju kratkog spoja između UART interfejsa na ploči i odgovarajućih pinova mikroprocesora pomoću multimetra. Nakon sprovedene procedure došlo se do informacije o korištenim UART kanalima kao i pinovima koji su u upotrebni. Ciljna platforma nudi šest RS-485 priključaka koji koriste šest UART kanala mikroprocesora. Korišteni kanali i konkretni pinovi mikroprocesora dati su u tabeli 2.1. Ono što se može napomenuti jeste da ciljna platforma koristi jedan UART kanal (UART0) u svrhe *debug* USB porta preko koga se serijskom vezom može pratiti proces učitavanja sistema te vršiti kontrola njegovog rada. Preko ovog priključka se takođe može vršiti programiranje *flash* memorijeciljne platforme, međutim ona ne dolazi sa potrebnim prekidačima da bi se uvela u taj režim rada. Zbog toga se moralo pristupiti drugom načinu prenosa aplikacije na ciljnu platformu, koji će biti kasnije pojašnjen detaljnije.

Tabela 2.1 - *Korišteni UART kanali i pinovi na ciljnoj platformi*

|  |  |  |
| --- | --- | --- |
| **Kanal** | **RX pin** | **TX pin** |
| UART1 | A0 | F10 |
| UART2 | A9 | A10 |
| UART3 | D3 | D2 |
| UART4 | D13 | D12 |
| UART6 | A4 | A5 |
| UART8 | A11 | A12 |

Kao što je već rečeno, poznavanje rasporeda pinova koji su iskorišteni je neophodno u fazi razvoja Linux operativnog sistema, odnosno za kreiranje strukture stabla uređaja. Ono što je još važno napomenuti prilikom razmatranja hardvera ciljne platforme jeste da što se tiče serijskog interfejsa MODBUS RSU protokola – RS-485, ciljna platforma sadrži 6 priključaka koji odgovaraju diferencijalnim A i B linijama RS-485. Signali sa ovih linija prolaze kroz odgovarajuću električnu mrežu da bi se doveli prema RX i TX linijama UART modula mikroprocesora. Električna mreža koja se nalazi između ima za ulogu da RS-485 diferencijalni signal (koji je većeg napona) obradi i konvertuje u +3.3V UART signal koji očekuje mikroprocesor. Prvi dio ove mreže nakon priključaka za RS-485 linije jeste sigurnosni i zaštitni dio koji se u slučaju ciljne platforme sastoji od redno vezanog osigurača i dva reda zaštitnih (TVS) dioda. Obje signalne linije RS-485 protokola (i A i B) prolaze kroz ovaj dio mreže, te idu na posebno integrisano kolo MAX13487E. Ovo kolo ima glavnu ulogu konverzije signala iz RS-485 oblika u signal koji se očekuje na UART linijama, i obrnuto. Dakle, kolo MAX13487E je RS-485 kolo primopredajnika zaduženo za konverziju oblika signala iz RS-485 u standardni UART, i obrnuto [4]. Treći i konačni korak obrade signala u datoj mreži jeste integrisano kolo CA-IS3722HS proizvođača *Chipanalog*. Ovo kolo je digitalno dvoulazno CMOS kolo koje služi za galvansku izolaciju ulaza i izlaza. Ovo kolo visokih performansi podržava brzinu prenosa podataka do 150 Mbps, specijalno namjenjeno u slučaju upotrebe za konverzije naponskih nivoa između serijskih protokola gdje su sa ulazne strane protokoli kao RS-485, CAN i slični [5]. U slučaju ciljne platforme, signali sa kola MAX13487E (sa pinova DI i RO) dovode se na pinove kola CA-IS3722HS gdje se galvanski izoluju sa pinovima na drugoj strani. Ovim se obezbjeđuje da pinovi koji imaju visoke naponske nivoe sa RS-485 strane galvanski izoluju i spuste na nivoe koje odgovaraju UART modulu mikroprocesora. Naravno važi i obrnut proces, kod koga se UART signali poslati sa mikroprocesora konvertuju u RS-485 kompatibilan signal preko iste ove mreže. Na slici 2.5 prikazana je ugrubo električna šema koja odgovara prethodno opisanoj mreži.

A computer screen shot of a yellow square

AI-generated content may be incorrect.  
Slika 2.5 – *Okvirna električna šema konverzije signala iz RS-485 u UART*

Prethodnom diskusijom pokrivena je hardverska analiza ciljne platforme. Naredni korak je priprema operativnog sistema koji će se izvršavati na ploči, a upravo to je tema narednog poglavlja.

# 3. Priprema operativnog sistema za ciljnu platformu

Kao operativni sistem (OS) izabran je Linux OS kao najčešće korišten operativni sistem u domenu ugrađenih računarskih sistema. Razvoj aplikacija koje se izvršavaju na Linux operativnom sistemu sa sobom nose veliku prednost u odnosu na *bare-metal* pristup razvoja, a to je portabilnost. Naime, aplikacije razvijane za neki operativni sistem mogu da se pokreću na bilo kojoj ciljnoj platformi, nezavisno od hardvera, sve dok taj hardver podržava izvršavanje Linux operativnog sistema na njemu i ima dovoljno resursa koje zahtjeva data aplikacija. Pored samog operativnog sistema, za razvoj aplikacija potrebno je na razvojnoj platformi imati odgovarajući "alat" - *toolchain*. *Toolchain* predstavlja skup alata neophodnih za razvoj i kreiranje izvršnih fajlova koji se mogu pokrenuti na ciljnoj platformi. Osnovno sadrži alat kroskompajler (*cross-compiler*), ali i neke druge alate. S obzrom na sve potrebno da bi se i ciljna i razvojna platforma pripremile za razvoj aplikacije, najbolji izbor jeste upotreba nekog *build* sistema koji na relativno jednostavan način može da generiše sve što je neophodno.

## 3.1. *Buildroot*

U slučaju ovog projektnog zadatka korišten je *build* sistem *buildroot*, odnosno, tačnije rečeno*, fork* repozitorijuma ovog *build* sistema koji je napravio proizvođač mikroprocesora, kompanija Nuvoton [6]. *Buildroot* je kompletan *build* sistem alat koji može da generiše sve neophodne artefakte za razvoj aplikacija na ugrađenom sistemu - *toolchain*, Linux jezgro, *bootloader,* korjeni fajl sistem (*root file system*) i sve ostalo. Kao i većina sistema, koristi *Make* alat za izgradnju.Ovaj alat je relativno jednostavniji *build* sistem pogodan za projekte manjih i srednjih razmjera namjenjenih za ugrađene računarske sisteme bazirane na Linux jezgru. Pored generisanja osnovnih artefakata potrebnih za učitavanje na ciljnu platformu i razvoj aplikacija, *buildroot* alat nudi mogućnost konfiguracije, kako samog Linux jezgra, tako i kompletnog sistema dodavanjem raznih biblioteka i paketa neophodnih za razvoj aplikacije [7]. Ove konfiguracije zasnivaju se na veoma poznatom *Kconfig* alatu.

Početak razvoja počinje kloniranjem pomenutog *fork*-a repozitorijuma *buildroot*-a lokalno na razvojnoj platformi [6]. S obzirom da je *fork* napravio proizvođač, u njemu se nalaze početne konfiguracije koje se mogu primjeniti tako da se konfiguracija *buildroot*-a olakša u smislu da se jednostavnom primjenom konfiguracije i izgradnjom već dobijaju artefakti Linux jezgra koji se mogu pokrenuti na ciljnoj platformi, bez potrebe da se ručno modifikuju neke konfiguracione opcije koje se tiču glavnog hardvera ciljne platforme (procesori, arhitektura procesora, podržane *floating point* jedinice, ...). S obzirom da se koristi *make* alat, prvi korak jeste učitavanje neke od početnih konfiguracija komandom *make <naziv\_početne\_konfiguracije>\_defconfig*. U slučaju ovog projektnog zadatka iskorištena je početna konfiguracija pod nazivom *nuvoton\_nuc980\_iot\_defconfig*. Nakon učitavanja početne konfiguracije, komandom *make menuconfig*, može se pristupiti detaljnijom konfiguraciji sistema. U slučaju ovog projektnog zadatka, u ovom dijelu konfiguracije odabrani su samo još neki dodatni paketi i biblioteke potrebni za izradu projektnog zadatka. Ove biblioteke i paketi navedeni su u nastavku:

* *libmodbus* (**BR2\_PACKAGE\_LIBMODBUS**) - jednostavna i široko poznata biblioteka koja implementira funkcionalnosti MODBUS protokola,
* *libjansson* (**BR2\_PACKAGE\_JANSSON**) - *lightweight* biblioteka čija je svrha jednostavno parsiranje i kreiranje *json* fajlova (kasnije će biti detaljnije pojašnjena uloga ovih fajlova u projektu),
* *openssh* (**BR2\_PACKAGE\_OPENSSH**) - biblioteka potrebna za realizaciju *ssh* konekcije, koristi se u fazi razvoja aplikacije za lako povezivanje razvojne i ciljne platforme preko mreže u cilju prenosa fajlova i udaljene kontrole ciljne platforme,
* *gpio* i *iio* alati (**BR2\_PACKAGE\_LINUX\_TOOLS\_GPIO** i **BR2\_PACKAGE\_LINUX\_TOOLS\_IIO**) - korisni alati koji nude pristup hardverskim pinovima razvojne platforme i njihovu kontrolu na visokom nivou iz korisničkog prostora jezgra.

Nakon ovih konfiguracija, pokretanjem komande *make* pokreće se izgradnja svih pomenutih artefakata. Po završetku procesa izgradnje na razvojnoj platformi su dostupni svi alati neophodni za razvoj aplikacija na ciljnoj platformi (*toolchain*), kao i slika jezgra, binarni fajl strukture stabla uredjaja, *bootloader*, korjeni fajl sistem i sve ostalo (ovi fajlovi nalaze se na putanji *output/images* iz lokacije *buildroot* foldera). Još je ostalo da se razjasni kako se generisani fajlovi mogu programirati na ciljnu platformu, tako da se omogući učitavanje Linux jezgra i rad sa platformom. U opštem slučaju (npr. razvojne ploče kompanije Nuvoton), programiranje memorije ploče sa ovim fajlovima može se izvršiti upotrebom programatorskog alata kompanije Nuvoton - *NuWriter*. Da bi se omogućilo programiranje mikroprocesora, odnosno odgovarajuće *flash* memorije povezane sa mikroprocesorom, dva BOOT pina ploče (PG1 i PG0) moraju biti povezana u odgovarajući režim. Taj režim je režim USB BOOT, gdje oba pina moraju biti na niskom logičkom nivou. Ovo podešavanje je hardversko, odnosno razvojne ploče tipično sadrže odgovarajuće prekidače kojima se konfigurišu naponski nivoi ova dva pina. Nakon što se pristupi USB BOOT režimu, ploča se može programirati korištenjem slike *bootloader*-a, Linux jezgra i binarnog fajla stabla uređaja (*.dtb* fajl) preko pomenutog alata za programiranje [8]. Nakon programiranja, BOOT pinovi se vraćaju u drugi režim (*boot* preko odgovarajuće memorije, *flash* ili *spi flash*) te se učitavaju novoprogramirane binarne slike *bootloader*-a i Linux jezgra.Takođe, moguća je i opcija učitavanja preko SD/eMMC kartice, međutim ciljna platforma nema hardversku podršku za istu. Takođe, ciljna platforma nema potrebne prekidače za podešavanje odgovarajućeg režima *boot*-a. Dakle, ponovo se nailazi na određene probleme koje samo po sebi nosi korištenje ciljne platforme razvijene u komercijalne svrhe. Da bi se prevazišao ovaj problem, primjenjena je sledeća ideja. Preko USB porta za *debug* prati se proces *boot*-ovanja ciljne platforme sa komercijalno razvijenim sistemom. Ovaj proces se sastoji iz dva glavna koraka. Prvo nastupa *bootloader* koji je u ovom slučaju vrlo poznat u oblasti ugrađenih sistema - *U-Boot*, nakon čega nastupa učitavanje Linux jezgra. Proces podizanja sistema može se prekinuti na pola, odnosno, može se prekinuti tokom izvršavanja *U-Boot*-a čime se pristupa *U-Boot* komandnoj konzoli. *U-Boot* je veoma moćan *bootloader* koji nudi razne mogućnosti tokom svog izvršavanja. Jedna od tih mogućnosti jeste konfiguracija mreže (podešavanje IP, MAC adrese, adrese servera, gejtveja i ostalo) i korišćenje nekih osnovnih protokola, kao što je *Trivial File Transfer Protocol* (*TFTP*). U *U-Boot* na ciljnoj platformi potrebno je podesiti varijable okruženja za IP adresu (*ipaddr*), adresu servera - razvojna platforma (serverip) i MAC adresu *emac* kontrolera (*ethaddr*). Na razvojnoj platformi potrebno je osposobiti TFTP server, te u folder na kome tftp server očekuje fajlove za prenos, kopirati sliku Linux jezgra (*uImage*) i binarnu sliku stabla uređaja (*.dtb* fajl). Nakon ovoga, moguće je iz *U-Boot*-a na ciljnoj platformi učitati date fajlove putem TFTP protokola komandom *tftp* u radnu memoriju, te izvršiti proces podizanja Linux jezgra komandom *bootm* [9]. Ovaj način podizanja operativnog sistema na ugrađenim računarskim sistemima poznat je i kao podizanje sistema preko mreže i iskorišten je kao najjednostavnije rješenje da se premosti prethodno opisani problem programiranja ciljne platforme.

## 3.2. Konfiguracija Linux jezgra, Linux rukovaoci uređaja, struktura stabla uređaja

Postupkom opisanim u prethodnoj podglavi dobijaju se svi potrebni fajlovi za podizanje Linux jezgra na ciljnoj platformi. Što se tiče samog Linux jezgra, *buildroot* alat koristi takođe specijalni *fork* Linux jezgra koji je takođe obezbjedila kompanija Nuvoton.Dato Linux jezgro se postupkom mrežnog podizanja sistema može bez problema izvršavati na ciljnoj platformi, čime se obezbjedio glavni uslov za izradu aplikacije projektnog zadatka. Međutim, ono o čemu se mora voditi računa jeste da tako generisano Linux jezgro možda ne podržava sav hardver ili periferije neophodne za realizaciju zahtjevanih projektnih zadataka. U drugoj glavi opisan je potreban hardver koji podrazumijeva dva Ethernet porta i šest UART kanala (pored, naravno, nekih dodatnih hardverskih modula kao što je npr. DMA kontroler). Da bi se omogućili, konfigurisali i uspješno koristili neophodni hardverski moduli neophodno je obratiti pažnju na dvije komponente Linux jezgra, a to su rukovaoci uređaja - drajveri (*drivers*) i struktura stabla uređaja (*device tree*).

### 3.2.1. Rukovaoci uređaja (*Linux Device Drivers*)

Rukovaoci uređaja su jedan od centralnih pojmova Linux operativnog sistema. Njihova uloga jeste da korisnicima (programerima i njihovim aplikacijama) obezbjedi jednostavan i dobro poznat interfejs kojim korisnička aplikacija može da interaguje sa određenim hardverom sistema na kome se izvršava Linux jezgro. Sva kompleksnost i pojedinosti datog hardvera skrivene su od korisničke aplikacije, ona samo koristi poznati interfejs (*API*) koji obezbjeđuje drajver. Drajveri su zaduženi da zahtjeve na visokom nivou koji dolaze od korisničkih aplikacija implementiraju te direktno interaguju sa hardverom za koji su zaduženi u svrhu izvršenja tih zahtjeva [10].

Omogućavanje odgovarajućih drajvera prilikom konfiguracije i izgradnje Linux jezgra omogućeno je takođe upotrebom *Kconfig* alata. Što se tiče ovog projektnog zadatka, potrebno je da u konfiguraciji omogućimo sve neophodne drajvere za Ethernet, kao i za UART komunikaciju. Koristeći *buildroot* alat, ovim konfiguracijama možemo pristupiti, i mijenjati ih, putem komande *make linux-menuconfig*. Potrebni drajveri koje treba omogućiti (ugraditi u sliku jezgra ili kao module) dati su u nastavku [11]:

* ICPlus PHYs (**CONFIG\_ICPLUS\_PHY**) - drajver za fizički sloj, neophodan za Ethernet komunikaciju,
* Nuvoton NUC980 Ethernet MAC 0 (**CONFIG\_NUC980\_ETH0**) - drajver prvog Ethernet porta,
* Nuvoton NUC980 Ethernet MAC 1 (**CONFIG\_NUC980\_ETH1**) - drajver drugog Ethernet porta i
* NUC980 UARTx Support (**CONFIG\_NUC980\_UARTx**) - drajveri za UART serijsku komunikaciju, pri čemu je 'x' broj UART kanala koji se koristi, a koji se nalaze u tabeli 2.1.

Rukovaoci uređaja su neophodni da bi korisničke aplikacije pristupile hardveru od interesa, međutim, oni ne mogu sami u potpunosti da obave posao. Naime, drajveri moraju nekako da dobiju informaciju o tome gdje su odgovarajuće periferije povezane sa mikroprocesorskom jedinicom i na koji način, kao i da dobiju informaciju o nekim osnovnim parametrima i konfiguracijama hardverskih elemenata sistema. Ove informacije drajveri dobijaju od drugog veoma bitnog dijela Linux sistema - strukture stabla uređaja (*device tree*).

### 3.2.2. Stablo uređaja (*Device Tree*)

Stablo uređaja predstavlja strukturu podataka i jezik koji služi za opis hardvera sistema. Kao što ime kaže, struktura izvornih fajlova (*.dts* fajlovi) sastoji se od čvorova (koji predstavljaju hardverske entitete sistema) koji se dodaju hijerarhijski u obliku stabla, onako kako bi se to očekivalo u hardverskoj hijerarhiji. Način kreiranja i pisanja čvorova za neki hardver ili periferiju koja je povezana na sistem, definisan je opštim pravilima koja se nazivaju *Device Tree Bindings (dt-bindings).* Jedan od glavnih ciljeva upotrebe ove strukture jeste da se drajverima uređaja omogući jedan uniforman i jednostavan način za konfiguraciju određenih parametara hardverskog entiteta kojim upravlja, te informacije o hardverskim vezama mikroprocesora i datog entiteta. Sve ove informacije drajver uređaja dobija preko njemu odgovarajućeg čvora u izvornom fajlu stabla uređaja (definisano *compatible* poljem). Ovim se ostiže elegantan način konfiguracije drajvera i hardvera tokom podizanja i rada samog Linux sistema, bez potrebe da se ovakve stvari "*hard* koduju" u izvorni kod drajvera, te mijenjaju svaki put kada treba neka promjena što bi takođe zahtjevalo ponovnu gradnju Linux jezgra [12].

Što se tiče ovog projektnog zadatka, neophodno je podesiti čvorove stabla uređaja koji se tiču Ethernet kontrolera i odgovarajućih UART modula. Svi ovi čvorovi kao i ostale bitne konfiguracije, već su kreirani u krovnom *nuc980.dtsi* fajlu za mikroprocesor NUC980 koji je obezbijedila kompanija Nuvoton. Na korisniku ostaje da kreira zaseban izvorni fajl stabla uređaja, u kom će da iskoristi kreirane čvorove, omogući ih kroz polje *status* i, ukoliko je to neophodno, podesi odgovarajuće pinove na koje su povezane periferije. U nastavku prvo će biti prikazan način na koji se preko strukture stabla uređaja daju informacije o pinovima. Ovo je moguće preko čvora koji se obično naziva *pinctrl*. Unutar ovog čvora se definišu podčvorovi koji odgovaraju periferijama, te unutar tih čvorova se kreiraju konačni čvorovi koji predstavljaju konfiguraciju pinova. U drugoj glavi razmotreni su pinovi na koje su povezani Ethernet portovi - pinovi E banke od E0 do E9 za prvi, odnosno F banka od F0 do F9 za drugi port. Ovakvu konfiguraciju možemo uočiti ako pogledamo *emac0* i *emac1* podčvorove čvora *pinctrl* što je prikazano na slici 3.1.

A screenshot of a computer code

AI-generated content may be incorrect.  
Slika 3.1 – *pinctrl podčvorovi za Ethernet kontrolere*

Sa slike se može uočiti da se koristi konstrukcija *nuvoton,pins* koja očekuje nizove konfiguracija pina od po 4 vrijednosti koje predstavljaju odgovarajuće osobine pina. Prva vrijednost je broj banke pina pri čemu se kreće od banke A (vrijednost 0) do banke G (vrijednost 6). Druga vrijednost je indeks pina iz date banke, pri čemu svaka banka ima ukupno 16 pinova (indeksi od 0 - 15). Treća opcija je vrlo važna i označava funkciju pina koja se može odabrati konceptom remapiranja. Konkretne vrijednost i funkcije pina vezane za njih dati su u dokumentaciji proizvođača [3]. Četvrta vrijednost su neke dodatne osobine konkretnog pina, i ova vrijednost se rijetko unosi [11]. Na slici 3.2 prikazan je primjer podčvora *uart1* čvora *pincrtl*.

A screenshot of a computer code

AI-generated content may be incorrect.  
Slika 3.2 – *Podčvor* *uart1* *čvora* *pinctrl*

Sa slike može se primjetiti da postoje neke predefinisane konfiguracije pinova, s obzirom na spomenute mogućnosti remapiranja koje su razmotrene u drugoj glavi. Ono što se može primjetiti jeste da nema definisane konfiguracije za onu koja je potrebna u slučaju hardverskih veza ciljne platforme (pogledati tabelu 2.1). U ovom slučaju, taj čvor sa željenom konfiguracijom pinova će biti neophodno dodati u izvorni fajl stabla uređaja koji se korisnički definiše, što će biti prikazano kasnije. Na sledećim slikama 3.3 i 3.4 su prikazani primjeri konkretnih čvorova Ethernet i UART kanala koji će biti proslijeđeni drajverima radi konfiguracije, i koji se mogu naći u pomenutom *.dtsi* fajlu.

A computer screen with white text

AI-generated content may be incorrect.  
Slika 3.3 – *Čvorovi Ethernet kontrolera*

A screenshot of a computer program

AI-generated content may be incorrect.  
Slika 3.4 – *Primjeri čvorova UART kanala*

Ono što ostaje da se uradi u glavnom *.dts* fajlu jeste da se odgovarajući čvorovi iz ovog fajla referensiraju, omoguće i po potrebi dodatno konfigurišu tako da odgovaraju stvarnom stanju hardvera na ciljnoj platformi. Konačno, na slici je dat isječak iz glavnog *.dts* fajla u kome su konfigurisani svi potrebni UART kanali koji se koriste na ciljnoj platformi. Sa slike se može primjetiti da je dodana odgovarajuća konfiguracija pinova koje koristi kanal UART1, pošto ista nije bila dostupna u *.dtsi* fajlu. Nakon toga su omogućeni potrebni UART kanali, pri čemu je za svaki kanal odabrana ona konfiguracija pinova koja odgovara stvarnim vezama na ciljnoj platformi. Kompletan *.dts* fajl sadrži dodatne čvorove koji su referensirani iz glavnog fajla i koji su onemogućeni (npr. I2C i SPI čvorovi) jer nisu potrebni za konkretan projekat.

Prethodno opisanom procedurom dobija se pripremljena konfiguracija Linux jezgra na kojoj je omogućen i podešen za rad sav zahtjevan hardver koji se nalazi na ciljnoj platformi. Do pokretanja *make* komande ostaje samo još jedan korak, koji nije obavezan, ali za konkretan sistem je preporučljiv.

A screenshot of a computer program

AI-generated content may be incorrect.  
Slika 3.5 – *Isječak glavnog .dts fajla*

### 3.2.3. Konfiguracija Linux jezgra za rad u realnom vremenu

Kao što je već rečeno, gejtvej je mrežni uređaj koji spada u grupu ugrađenih računarskih sistema. Ovakvi uređaji često imaju zahtjeve za rad u realnom vremenu i često se smatraju kao ugrađeni sistemi koji rade u realnom vremenu. Zbog toga se kao poslednji korak u pripremi Linux operativnog sistema izvodi njegova konfiguracija za rad u realnom vremenu, poznata kao “pečiranje” Linux jezgra za rad u realnom vremenu (*Real-time patch*). Ova procedura podrazumijeva preuzimanje dostupnog fajla zakrpe (*patch file*) dostupnog onlajn, te njegovu primjenu na fajlove Linux jezgra primjenom komande *patch -p1* unutar korjenog foldera Linux jezgra [13]. Treba voditi računa da fajl zakrpe mora odgovarati verziji Linux jezgra koji se izgrađuje (u ovom slučaju verzija je 5.10.140). Fajl zakrpe modifikuje razne konfiguracione fajlove unutar izvornog koda jezgra čime omogućava njegovu konfiguraciju kao sistem za rad u realnom vremenu. Ove konfiguracije mijenjaju mogućnosti nekih standardnih biblioteka (na primjer biblioteke *pthread*, od koje uveliko zavisi biblioteka *libiec60870*), ali ono što je najvažnije, podešavaju raspoređivač Linux jezgra tako da podrži *hard* zahtjeve za rad u realnom vremenu. Nakon primjene zakrpe, potrebno je samo omogućiti konfiguracionu opciju **CONFIG\_PREEMPT\_RT** čime je Linux jezgro potpuno prilagođeno za rad u realnom vremenu.

Nakon ove procedure, preostaje da se ponovo izgradi Linux jezgro upotrebom komande *make linux-rebuild*, te konačno izgradnja svih fajlova upotrebom *buildroot* alata pokretanjem komande *make*. Novonastala slika kernela i *.dtb* fajl mogu se učitati preko mreže i iskoristiti za podizanje jezgra na ciljnoj platformi. Ovim se završava sva procedura generisanja neophodnih resursa za razvoj aplikacije, kako na razvojnoj, tako i na izvršnoj platformi.

# Literatura

[1] https://gateway-iot.com/modbus-to-iec104-p00426p1.html, posjećeno: 20. 2. 2025.

[2] <https://www.nuvoton.com/products/microprocessors/arm9-mpus/nuc980-industrial-control-iot-series/>, posjećeno: 20. 2. 2025.

[3] NUC980 Series Techical Reference Manual, *Nuvoton*, Jan. 07, 2020.

[4] MAX13487E/MAX13488E: Half-Duplex RS-485-Compatible Transceiver withAutoDirection Control Data Sheet, *Analog Devices*, Rev 3., Aug. 2024.

[5] CA-IS372x High-Speed Dual-Channel Digital Isolators, *Chipanalog Microelectronics*, Rev 1.08, Dec. 17, 2024.

[6] https://github.com/OpenNuvoton/MA35D1\_Buildroot, posjećeno: 22. 2. 2025.

[7] https://buildroot.org/, posjećeno: 22. 2. 2025.

[8] NUC980 NuWriter User Manual, *Nuvoton*, Apr. 22, 2019.

[9] https://www.u-boot.org/, posjećeno 22. 2. 2025.

[10] Linux Device Drivers, *Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman*, O'Reilly Media Inc. February 2005.

[11] NUC980 Linux 5.10 BSP User Manual, *Nuvoton*, Sep. 1, 2023.

[12] https://docs.kernel.org/devicetree/usage-model.html, posjećeno: 22. 2. 2025.

[13] <https://cdn.kernel.org/pub/linux/kernel/projects/rt/5.10/>, posjećeno 22. 2. 2025.