Analiza i usporedba performansi ugradbenog Linux operacijskog sustava

Petrović, Tomislav

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ANALIZA I USPOREDBA PERFORMANSI UGRADBENOG LINUX OPERACIJSKOG SUSTAVA

Diplomski rad

Tomislav Petrović

Osijek, 2019.

Sadržaj

1. UVOD

Ugradbeni računalni sustav (engl. *embedded systems*) predstavlja kombinaciju računalnog hardvera i softvera, te ima mogućnost dodavanja dodatnih dijelova, mehaničkih ili elektroničkih s ciljem obavljanja specifične zadaće. Ugradbeni računalni sustav je često sastavni dio nekog većeg tehničkog sustava. Primjeri ovakvih sustava su moderni automobili, kamioni i slično. Kod ovakvih primjera jedan ugradbeni računalni sustav kontrolira rad kočnica, dok drugi prikazuje određene informacije na nadzornoj ploči itd. Dizajn ugradbenog računalnog sustava za obavljanje specifične zadaće u izravnoj je suprotnosti s onim kod osobnog računala koje je namijenjeno za različite zadaće.

Ugradbeni računalni sustavi su najčešće ograničeni veličinom, cjenovno moraju biti prihvatljivi pa su zbog toga najčešće skromnih računalnih mogućnosti, niske potrošnje, te trebaju osigurati stabilan i pouzdan rad. Ugradbeni računalni sustavi često upravljaju kritičnim procesima pa zahtijevaju posebnu programsku podršku [1]. Hardver ugradbenog računalnog sustava može biti zasnovan na mikroupravljaču ili mikroprocesoru, ali u oba slučaja integrirani krug nalazi se u središtu sustava koji je općenito odgovoran za operacije u stvarnom vremenu [2]. Mikroprocesor i mikroupravljač vizualno se ne moraju razlikovati, dok mikroprocesor implementira samo središnja procesorska jedinica (engl. *Central Processing Unit* - CPU) te zahtijeva dodavanje drugih komponenti kao što su memorijski čipovi. Mikroupravljač je dizajniran kao samostalni sustav, te on uključuje CPU, memoriju, periferne uređaje kao što su *flash* memorija, RAM, veći broj različitih ulazno/izlaznih portova kao što su serijski port, pulsno širinsko moduliran izlaz (eng. *Pulse Width Modulation* – PWM), analogno digitalni pretvornik (eng. *Analog to Digital Converter* - ADC), digitalno analogni pretvornik (eng. *Digital to Analog Converter* - DAC) koji su ključni za ugradbeni računalni sustav jer omogućavaju osnovnu funkcionalnost i interakciju s okolinom [1].

Softver ugradbenih računalnih sustava može se definirati kao niz programskih instrukcija pomoću kojih sustav obavlja zadanu zadaću. Softver ugradbenih računalnih sustava sastoji se od operacijskog sustava (engl. *Operating system* - OS) koji uređaju omogućava izvršavanje zadanih poslova i korisničkih aplikacija koje definiraju izvršavanje specifičnih zadataka. Ovisno o primjeni ugradbenog računalnog sustava postoje različite korisničke aplikacije u telekomunikacijama, automobilima, digitalnoj potrošačkoj elektronici i slično. Operacijski sustav predstavlja skup osnovnih programa koji se nalaze između korisnika i računala.

Operacijske sustave možemo podijeliti u tri skupine: operacijski sustavi opće namjene, operacijski sustavi posebne namjene i prilagođeni operacijski sustavi opće namjene. Prilikom korištenja operacijskog sustava opće namjene dostupne su određene tehnologije, protokoli, standardi, te potrebni alati. Operacijski sustavi opće namjene su oni sustavi koji mogu osigurati rad s različitim programima, te može omogućiti interaktivnu komunikaciju s korisnikom. Operacijski sustav posebne namjene je sustav koji je prilagođen za rad u strogim vremenskim ograničenjima, te imaju različita svojstva u odnosu na operacijske sustave opće namjene. Jedno od ključnih svojstava je to što operacijski sustav opće namjene ne sadrži odgovarajuće mehanizme vremenske određenosti i značajno odudara dimenzijama i cijenom od zahtijeva ugradbenih računalnih sustava. Prilagođeni operacijski sustavi predstavlja operacijski sustav opće namjene na koji su primijenjene odgovarajuće modifikacije kako bi se omogućio deterministički rad operacijskog sustava. Primjer ovog sustava je operacijski sustav Linux operacijski sustav za rad u stvarnom vremenu (engl. *Real-Time Linux*) koji predstavlja jezgru operacijskog sustava s odgovarajućim dodatkom na izvorni kod Linux jezgre [19]. U daljnjem radu Linux operacijski sustav za radu stvarnom vremenu označava se LinuxRTOS. Ugradbeni sustavi za rad u stvarnom vremenu su tip računala koji reagira na vanjske događaje ili ulazne podražaje pravovremeno (unutar konačnog i određenog vremena). Operacijski sustavi za rad u stvarnom vremenu namijenjeni su za posluživanje aplikacija u stvarnom vremenu koje obrađuju podatke kako dolaze, obično bez kašnjenja. Stvarno vrijeme podrazumijeva uključivanje ograničenja koja se odnose na vrijeme odziva, vrijeme početka i vrijeme završetka tj. vrijeme potrebno da se odgovori da događaj [5]. Ugradbeni sustav za rad u stvarnom vremenu (engl. *Real Time Operating System* - RTOS) mora generirati ispravne računske odgovore na događaje (funkcionalna ograničenja), a odgovori ili rezultati moraju biti izrađeni u unaprijed određenom vremenu (vremenska ograničenja). Linux je najpoznatiji i najčešće korišteni operacijski sustav otvorenog koda. U odnosu na ostale operacijske sustave, Linux je operacijski sustav otvorenog koda te je svakom korisniku besplatan i dostupan za pregled i uređivanje. Prednost korištenja Linux operacijskog sustava u ugradbenim računalnim sustavima je to što sadrži podršku za velik broj različitih sklopovlja, besplatan je za korištenje, softver otvorenog koda i slično. Korištenje Linux operacijskog sustava u ugradbenim računalnim sustavima omogućava izvršavanje složenih zadataka uz pomoć postojećih programa bez da su potrebne dodatne nadogradnje funkcionalnosti, što predstavlja glavnu prednost korištenja Linux operacijskog sustava u odnosu na druge operacijske sustave. Problem Linux operacijskog sustava za

primjenu u ugradbenim računalnim sustavima koji zahtijevaju rad u stvarnom vremenu je nemogućnost reagiranja na događaj u strogo definiranom vremenskom intervalu.

Kako ugradbeni projekti zahtijevaju operativni sustav za rad u stvarnom vremenu, programeri Linuxa često koriste PREEMPT-RT zakrpu Linux jezgre koja omogućava rad u stvarnom vremenu. Linux za rad u stvarnom vremenu (engl. *RT-Linux*) razvili su V. Yodaiken i M. Barabanov 1996. godine što su pridružili Linux jezgri te tako omogućili izvršavanje zadataka s vremenskih ograničenjima. Trenutno postoje različiti pristupi za korištenje Linuxa za rad u stvarnom vremenu. Jedan od načina je korištenje PREEMPT-RT zakrpe koja za cilj ima stvaranje predvidljivo i determinirano okruženje pretvarajući Linux jezgru u održivu platformu za rad u stvarnom vremenu. Glavna prednost ovog načina je mogućnost korištenja standardnih Linux alata i odgovarajućih biblioteka.

Prilikom izrade diplomskog rada napravljena je usporedba performansi dvije inačice Linux operacijskog sustava: Linux operacijski sustav za rad u stvarnom vremenu u daljnjem tekstu LinuxRTOS i LinuxOS. Navedene inačice izgrađene su na temelju izvornog koda Linux jezgre (eng. *Linux kernel*) uz pomoć Buildroot alata za jednostavniju izgradnju Linux operacijskih sustava za Raspberry Pi 3 Model B razvojnu ploču. Usporedba performansi temelji se na odgovarajućim mjerenjima koja su pisana u C programskom jeziku, a prikazana su u tablicama i pomoću grafova.

Ovaj rad podijeljen je u pet poglavlja. U drugom poglavlju opisani su operacijski sustavi za rad u stvarnom vremenu, a poseban osvrt dan je na Linux operacijski sustav koji se koristi prilikom izrade diplomskog rada. U trećem poglavlju predstavljeno je vlastito rješenje, potrebne informacije o Raspberry Pi 3 Model B razvojnoj ploči i Buildroot alatu, te informacije o izgradnji Linux jezgre za Raspberry Pi 3 Model B razvojnu ploču pomoću Buildroot alata. U četvrtom poglavlju opisana su testiranja koja su podijeljena u dvije grupe: testiranja na standardnom Linux operacijskom sustavu i testiranja na Linux operacijskom sustavu za rad u stvarnom vremenu. Peto poglavlje sadrži zaključak.

2. SUSTAVI ZA RAD U STVARNOM VREMENU

Ugradbeni sustavi za rad u stvarnom vremenu postoje gotovo u svakom aspektu svakodnevnog života, a mogu se pronaći u različitim mobilnim uređajima, automobilima, kuhinjskim aparatima te takve sustave smatramo jednostavnijim ugradbenim sustavima za rad u stvarnom vremenu. Osim gore navedenih primjera postoje i složeniji primjeri, a neki od njih su: sustav kontrole zračnog prometa, vojni oružani sustavi, robotika i automatizacija. Ugradbeni sustavi za rad u stvarnom vremenu imaju složen skup karakteristika i zadaća koje zahtijevaju posebnu programsku podršku. Ugradbeni sustavi za rad u stvarnom vremenu reagiraju na različite događaje iz vanjskog svijeta, pridržavajući se određenih zahtjeva koje postavlja okolina u kojoj djeluju. Ispravnost sustava ne ovisi samo o rezultatima izračunavanja, nego i o vremenu u kojem su rezultati dobiveni. Najvažnija i najsloženija karakteristika ugradbenih sustava za rad u stvarnom vremenu je da oni moraju primiti i reagirati na skup vanjskih podražaja unutar kritičnih vremenskih ograničenja [5].

2.1. Tipovi sustava za rad u stvarnom vremenu

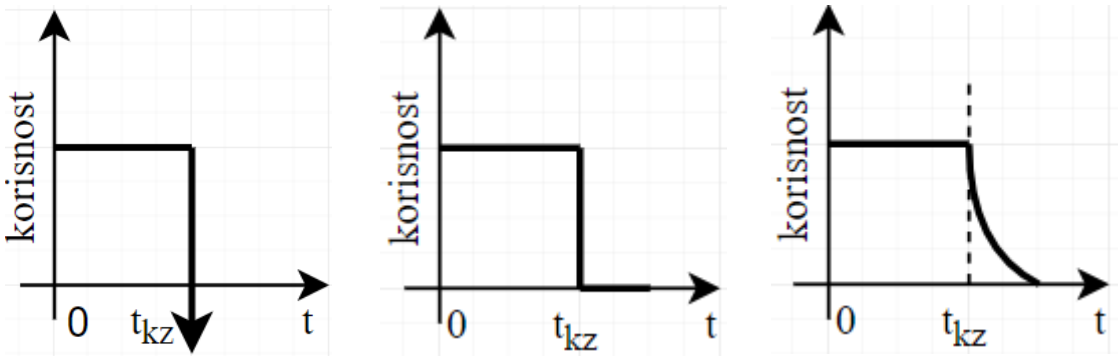
Osnovna podjela sustava za rad u stvarnom vremenu temelji se na očekivanjima sustava, odnosno posljedicama nepridržavanja zadanih vremenskih ograničenja. Prema tim uvjetima sustavi za rad u stvarnom vremenu dijele se na:

a) sustav sa strogim vremenskim zahtjevima (engl. *hard deadline*) – predstavlja sustav koji je čisto deterministički i vremenski ograničen sustav. Sustav u kojem vrijeme nužnog završetka nakon kojeg rezultat nije koristan može izazvati velike posljedice kao što su gubitak života, ekološka katastrofa i slično. Primjeri sustava sa strogim vremenskim zahtjevima su sustav kontrole leta, sustav za usmjeravanje projektila i slično. Na slici 2.1. a) prikazana je korisnost sustava za rad u stvarnom vremenu sa strogim vremenskim zahtjevima pri čemu x-os predstavlja vrijeme izvođenja, y-os korisnost sustava, a t_{kz} predstavlja rok izvršavanja određene zadaće [6].

b) sustav s čvrstim vremenskim zahtjevima (engl. *firm deadline*) – predstavlja sustav kojem su vremena nužnog završetka bitna, ali sustav će raditi ispravno i nakon predviđenog vremena. Kod ovog tipa sustava, ako odgovor nije dostavljen u roku neće doći do katastrofalnog rezultata samo će biti vidljivo veliko kašnjenje. Primjeri sustava sa čvrstim vremenskim zahtjevima: upravljačka kutija TV-a za biranje videozapisa na

zahtjev korisnika (engl. *Set Top Box* - STB), DVD uređaji, sustavi vremenskog nadzora i slično. Na slici 2.1. b) prikazana je korisnost sustava za rad u stvarnom vremenu s čvrstim vremenskim zahtjevima pri čemu x-os predstavlja vrijeme izvođenja, y-os korisnost sustava, a t_{kz} predstavlja rok izvršavanja određene zadaće [19].

c) sustav s ublaženim vremenskim zahtjevima (engl. *soft deadline*) – predstavlja sustav koji neće u potpunosti zakazati ako se neki od zadataka ne izvrši u određenom vremenskom roku. Kod sustava s ublaženim vremenskim zahtjevima dozvoljeno je poneko prekoračivanje, ali se time smanjuje korisnost sustava. Primjer sustava s ublaženim vremenskim zahtjevima je računalo za upravljanje grijanjem. Na slici 2.1.c) prikazana je korisnost sustava za rad u stvarnom vremenu s ublaženim vremenskim zahtjevima pri čemu x-os predstavlja vrijeme izvođenja, y-os korisnost sustava, a t_{kz} predstavlja rok izvršavanja određene zadaće [19].



a) b) c)

Sl. 2.1. *Tipovi sustava za rad u stvarnom vremenu. (a)čvrsti vremenski zahtjevi, (b)strogi vremenski zahtjevi i (c)ublaženi vremenski zahtjevi [19].*

Na temelju načina na koji se događaji u stvarnom vremenu ponavljaju kroz vremensko razdoblje, moguće ih je klasificirati u tri glavne kategorije:

a) periodični događaji – predstavljaju događaje koji se pojavljuju u istim vremenskim razmacima ili se u istim vremenskim intervalima pojavljuje samo jedan takav događaj. Uobičajeni periodični događaji su obrada podataka senzora i redovito ažuriranje trenutnog stanja sustava za rad u stvarnom vremenu. Periodični događaji obično se koriste u aplikacijama za kontrolu i obradu signala koji imaju čvrste vremenske zahtjeve.

b) aperiodični događaji – predstavljaju događaje koji dolaze u nepravilnim vremenskim intervalima. Aperiodični događaji koriste se za obradu zahtjeva slučajnog događaja kao što su primjerice zahtjevi operatera.
c) sporadični događaji – predstavljaju događaje koji se rijetko pojavljuju. Kod sporadičnih događaja potrebno je imati minimalno vremensko ograničenje, a ako minimalno vremensko ograničenje nije postavljeno, nemoguće je jamčiti da će uvijek biti ispunjen rok sporadičnog događaja.

2.2. Operacijski sustav

Operacijski sustav djeluje kao posrednik između korisnika računala i računalnog hardvera. Svrha operacijskog sustava je pružiti okruženje u kojem korisnik može izvršavati programe na prikladan i učinkovit način. Operacijski sustav predstavlja softver koji upravlja hardverom računala. Operacijski sustav brine se za raspodjelu resursa i usluga, poput memorije, procesora, uređaja i informacija. Operacijski sustav prema tome uključuje programe za upravljanje resursima kao što su upravljač prometa, planer, modul za upravljanje memorijom, datotečni sustav. Postoje nekoliko vrsta operacijski sustava, a jedni od njih su [21]:

- operacijski sustav s grupnom obradom (engl. *Batch Operating System*) – predstavlja vrstu operacijskog sustava koji ne komunicira izravno s računalom. Za komunikaciju postoji operator koji uzima slične poslove s istim zahtjevima i grupira ih u skupine. Problemi kod sustava s grupnom obradom su nedostatak interakcije između korisnika i posla.
- operacijski sustav dijeljenja vremena (engl. *Time-Sharing Operating System*) – dijeljenje vremena je tehnika koja mnogim korisnicima omogućava istovremeno korištenje određenog računalnog sustava. Prednosti operacijskih sustava dijeljenja vremena su izbjegavanje dupliciranje softvera, te smanjuje vrijeme neaktivnosti centralne procesne jedinice, dok su nedostatci operacijskog sustava dijeljenja vremena smanjena pouzdanost, pitanje sigurnosti i integriteta korisničkih podataka i programa.
- distribuirani operacijski sustav (engl. *Distributed Operating System*) – sustavi koji koriste više centralnih procesora za posluživanje više aplikacija u stvarnom vremenu i više korisnika. Prednosti distribuiranih operacijskih sustava su bolja usluga kupcima, smanjeno kašnjenje u obradi podataka, smanjeno opterećenje na glavnom računalu.

- mrežni operacijski sustav (engl. *Network Operating System*) – sustav koji se pokreće na poslužitelju i pruža poslužitelju mogućnost upravljanja podacima, korisnicima, grupama i drugim mrežnim funkcijama. Prednosti mrežnog operacijskog sustava su stabilni centralni poslužitelji, laka nadogradnja na nove tehnologije, dok su nedostatci veliki troškovi kupnje i pokretanja poslužitelja, potrebno je redovito održavanje i ažuriranje.
- operacijski sustav za rad u stvarnom vremenu – predstavlja sustav za obradu podataka gdje je vremenski interval za obradu podataka i reagiranje vrlo mal, vremenski interval naziva se još vremenskim odzivom. Kod ovog tipa operacijskog sustava vrijeme odziva je znatno manje u usporedbi s mrežnim operacijskim sustavom. Operacijski sustavi za rad u stvarnom vremenu koriste se kada postoje strogi vremenski zahtjevi s obzirom na zadaću sustava.

2.3. Raspoređivanje poslova unutar složenog upravljačkog programa

Raspored poslova na zadatke u stvarnom vremenu u osnovi odnosi se na utvrđivanje redoslijeda po kojem se različiti zadatci trebaju preuzeti na izvršavanje. Zadatak predstavlja manji dio složenog upravljačkog programa koji se brine za odgovarajući vanjski proces. Planiranje (engl. s*cheduling*) predstavlja funkciju operacijskog sustava koji upravlja redoslijedom zadataka koji će se pokrenuti na CPU. Zadatci u ugradbenim sustavima za rad u stvarnom vremenu moraju se izvršiti prije definiranog vremena. Vrste zadataka koji se mogu uništiti nakon pokretanja stvaraju potrebu za korištenjem algoritama za nepredviđeno planiranje i za odabir zadatka koji će optimizirati učinkovitost sustava. Svaki operacijski sustav oslanja se na jedan ili više planera zadataka kako bi pripremio raspored izvršenja različitih zadataka koje je potrebno izvoditi. Planer je dio jezgre koji je odgovoran za odlučivanje koji zadatak treba izvršiti u koje vrijeme [19]. Jezgra može obustaviti i kasnije nastaviti zadatak više puta tijekom radnog vijeka zadatka. Svaki postupak raspoređivanja planira se u skladu s jednim od sljedećih načina raspoređivanja [25]:

- SCHED_FIFO – predstavlja algoritam kod kojeg zadaci nemaju vremenske rokove, a pokreću se prema višem prioritetu u stvarnom vremenu (engl. *first in first out* -FIFO). Više zadataka SCHED_FIFO istog prioriteta izvršavaju se na način kada je zadatak obavljen prebacuje se na kraj čekanja i slijedi sljedeći zadatak.

- SCHED_RR – predstavlja identičan SCHED_FIFO algoritam samo što se svaki zadatak može pokrenuti sve dok se ne potroši unaprijed određeni vremenski interval (engl. *round – robin* – RR). Odnosno SCHED_RR je SCHED_FIFO s odvojenim vremenskim intervalima.
- SCHED_OTHER – predstavlja standardni Linux planer vremena koji je namijenjen svim zadacima koji ne zahtijevaju posebne mehanizme u stvarnom vremenu. SCHED_OTHER se koristi samo kada je statički prioritet jednak 0. Zadatak koji se pokreće odabran je s popisa 0 statičkog prioriteta na temelju dinamičkog prioriteta koji je određen samo unutar ovog popisa. Dinamički prioritet temelji se na lijepoj vrijednosti (engl. *nice value*).

2.4. Linux operacijski sustav

Linux je besplatni operacijski sustav otvorenog koda temeljen na UNIX-u koji je 1991. godine stvorio Linus Torvalds. Korisnici mogu mijenjati i stvarati varijacije izvornog koda poznatije kao distribucije. Najčešće se koristi kao poslužitelj, ali naravno Linux operacijski sustav se koristi i u stolnim računalima, pametnim telefonima, igraćim konzolama i slično. Postoje različite prednosti Linuxa i otvorenog koda u ugradbenim sustavima, a neke od njih su [23]:

- mogućnost ponovnog korištenja softverskih komponenti,
- otvoreni izvorni ekosustav već sadrži brojne komponente za uobičajene zahtjeve, od hardverske podrške do mrežnih protokola, različitih biblioteka, grafike i slično,
- širenjem upotrebe hardverskog uređaja, protokola ili karakteristika, velika je vjerojatnost da je komponenta otvorenog koda već dostupna,
- omogućava brzo dizajniranje i razvoj složenih komponenti na temelju postojećih komponenti,
- nema potrebe za razvijanjem dodatnih jezgri operacijskog sustava, TCP/IP stoga, USB stoga ili drugih biblioteka
- omogućuje usredotočenost na dodanu vrijednost komponente.

Kako je Linux operacijski sustav besplatan softver, moguće ga je besplatno kopirati na veći broj uređaja. Ako ugradbeni sustav koristi samo besplatni softver, troškove licenci moguće je smanjiti na nulu. Kod Linux operacijskog sustava također su besplatni i razvojni alati, osim u slučaju odabira komercijalne verzije ugradbenog Linux operacijskog sustava. Korištenjem

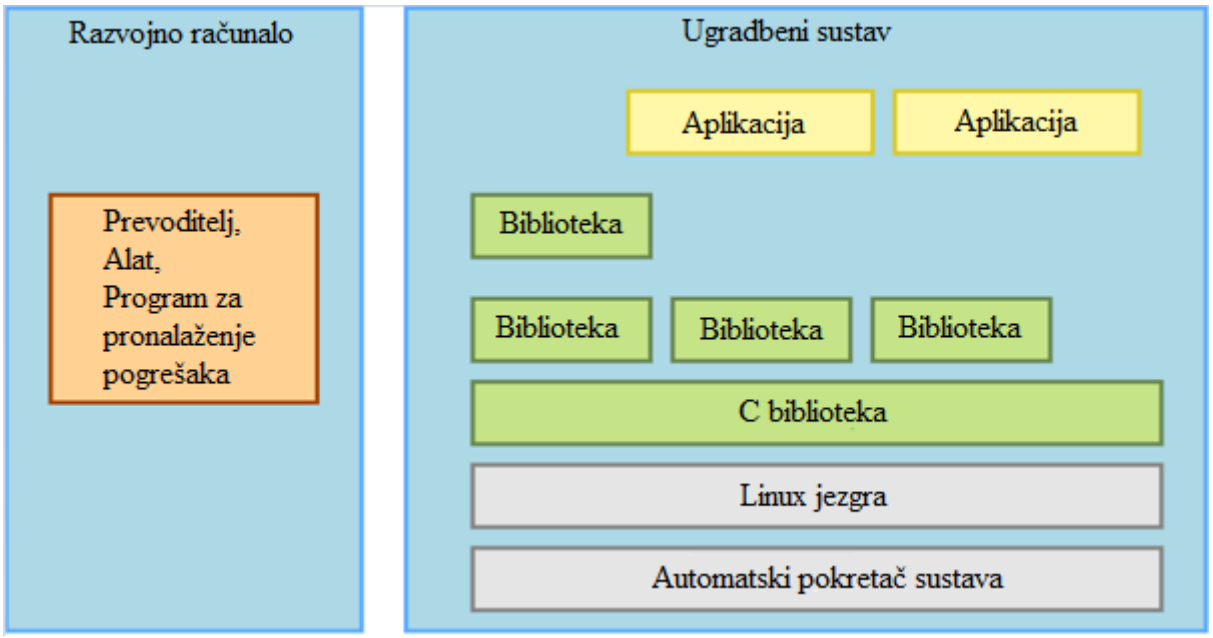
Linux operacijskog sustava otvorenog koda stvara se mogućnost upravljanja izvornim kodom svake komponente u sustavu. Takvim načinom korištenja omogućavaju se neograničene izmjene, promjene, podešavanja, ispravljanje pogrešaka i slično. Linux operacijski sustav omogućava potpunu kontrolu nad softverskim dijelom sustava. Mnoge komponente otvorenog koda koriste se u milijunima sustava, te često takve komponente imaju veću kvalitetu od onoga što odgovarajuća kompanija može proizvesti. Linux jezgra i većina ostalih komponenti ovisnih o arhitekturi podržavaju širok spektar 32 i 64-bitnih arhitektura, a neke od njih su [22]:

- x86 i x86-64 koji se nalaze na osobnim računalima, ali i ugradbenim sustavima (multimedijski, industrijski),
- ARM, sa stotinama različitih sustava u procesoru (engl. *System on a Chip* - SoC),
- RiscV,
- PowerPC i Microblaze za Xilinx FPGA.

Osim navedenih arhitektura podržane su i arhitekture s i bez jedinice za upravljanje memorijom (engl. *Memory Management Unit* - MMU). Linux operacijski sustav nije dizajniran za male mikroupravljače. Osnovni Linux operacijski sustav može raditi unutar 8MB RAM-a, ali malo realističniji sustav obično zahtjeva 32 MB RAM-a. Osnovni Linux operacijski sustav može raditi unutar 4MB prostora, ali obično je potrebno i više prostora. Linux jezgra ima podršku za više komunikacijskih sabirnica kao što su: I2C, SPI, CAN, SDIO, USB. Također Linux jezgra ima i punu podršku za mrežnu komunikaciju: Ethernet, Wifi, Bluetooth, CAN, IPv4, IPv6, TCP, UDP i slično [22].

Na slici 2.2. prikaza je arhitektura ugradbenog Linux operacijskog sustava. Na slici su prikazane komponente sustava, a to su [23]:

- prevoditelj za unakrsnu izgradnju jezgre (engl. *Cross Compiler*) – služi za izgradnju jezgre za odgovarajuću platformu na različitoj od one na kojoj se razvija,
- automatski pokretač sustava (engl. *Bootloader*) – odgovoran za osnovnu inicijalizaciju, učitavanje i izvršavanje jezgre,
- Linux jezgra – sadrži upravljanje procesima i memorijom, mrežni stog, upravljački programi uređaja,
- c biblioteka – sprega između jezgre i aplikacija u korisničkom prostoru,
- biblioteke i aplikacije.



Sl. 2.2. *Prikaz arhitekture Linux ugradbenog sustava [22].*

Prilikom korištenja Linux operacijskog sustava u pametnim telefonima i modernim automobilima, Linux operacijski sustav izvršava specifične zadatke, gdje se svaki zadatak izvršava u odgovarajućem trenutku. Kako bi se svaki zadatak izvršio u odgovarajućem trenutku potrebno je imati planere zadataka. Kada se govori o raspoređivanju zadataka kod Linux operacijskog sustava poznato je da Linux jezgra podržava odgovarajuće raspoređivače koji raspoređuju zadatke prema prioritetu zaprimanja. Linux operacijski sustavi podržavaju različite algoritme raspoređivanja. Linux OS koristi sljedeće algoritme raspoređivanja, a to su SCHED_FIFO, SCHED_RR, SCHED_OTHER koji su objašnjeni u prethodnom poglavlju, te SCHED_BATCH i SCHED_IDLE algoritme. SCHED_BATCH algoritam može se upotrebljavati samo kod statičkog prioriteta 0. Ovo pravilo je slično kao i kod SCHED_OTHER algoritma po tome što se planira postupak prema dinamičkom prioritetu. Razlika je u tome što će ovo pravilo uzrokovati da planer pretpostavlja da je zadatak intenzivan na CPU, te će na temelju toga planer primijeniti odgovarajuću tzv. kaznu, a zbog toga raspored izvođenja zadataka nije zadovoljen. SCHED_IDLE algoritam raspoređivanja koristi se samo kod statičkog prioriteta 0, ali kod njega lijepa vrijednost nema utjecaja na redoslijed izvođenja zadataka. Ova vrsta algoritma raspoređivanja namijenjena je za pokretanje zadataka s niskim prioritetom [28].

Osim algoritama za raspoređivanje zadataka prilikom izrade diplomskog rada bitan pojam predstavlja i latencija. Točna definicija latencije ovisi o sustavu koji se promatra. Latencija je

vremenski interval između podražaja i odgovora, odnosno vremensko kašnjenje između dva uzroka i učinka neke fizičke promjene koja se promatra u sustavu. Latencija predstavlja posljedicu ograničene brzine kojom se može širiti bilo koja fizička interakcija. Latencija se odnosi vremenski interval ili kašnjenje gdje komponenta sustava čeka da druga komponenta sustava izvrši svoj zadatak.

2.5. Linux operacijski sustav za rad u stvarnom vremenu - RTOS

Iako Linux nije operacijski sustav za rad u stvarnom vremenu, dostupne su različite dodatne opcije koje mogu pridonijeti stvaranju Linux operacijskog sustava za rad u stvarnom vremenu. Linux operacijski sustav pripada Unix obitelji, a glavni cilj Linux operacijskog sustava je maksimalna iskorištenost resursa. U početku Linux operacijski sustav nije bio pogodan za rad u stvarnom vremenu zbog nemogućnosti odgovaranja na vanjske događaje u određenom vremenskom intervalu. S vremenom Linux operacijski sustav razvija dva glavna pristupa koja postavljaju zahtjeve za rad u stvarnom vremenu. Prvi pristup predstavlja poboljšanje Linux jezgre i korištenje PREEMPT-RT projekta tako da odgovara zahtjevima u stvarnom vremenu. Drugi pristup predstavlja dodavanje dodatnog sloja ispod Linux jezgre koji će podnijeti sve zahtjeve u stvarnom vremenu tako da ponašanje Linux operacijskog sustava ne utječe na zadatke u stvarnom vremenu. Ovaj pristup preuzeli su RTLinux, RTAI i Xenomai [22].

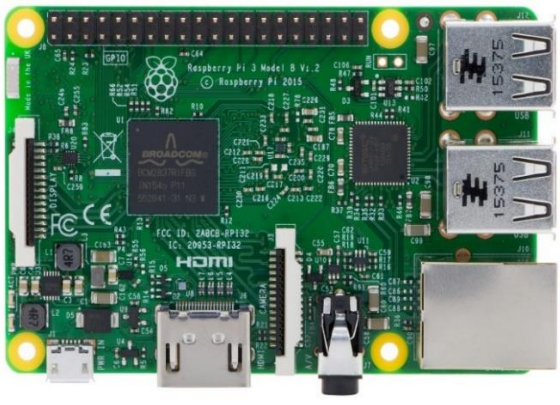
Izrada ovog diplomskog rada temelji se na izgradnji Linux jezgre korištenjem PREEMPT-RT projekta koji zahtjeva kompatibilnost razvojne ploče i glavne Linux jezgre. Starije inačice Linux operacijskog sustava ne podržavaju PREEMPT-RT projekt. Za postavljanje Linux operacijskog sustava za rad u stvarnom vremenu potrebno je preuzeti najnoviju inačicu PREEMPT-RT projekta s pouzdanog izvora. Nakon preuzimanja projekta potrebno je unutar konfiguracijskog dijela Linux jezgre postaviti parametre za rad u stvarnom vremenu, te pokrenuti izgradnju Linux operacijskog sustava. Ovaj projekt predstavlja nova poboljšanja operacijskog sustava kako bi se smanjilo i maksimalno i prosječno vrijeme odziva Linux jezgre. Kako je poznato da Linux operacijski sustav nije *preemptible* što znači da se jezgra ne može prekinuti, što može uzrokovati latencije. Upravo korištenje PREEMPT-RT projekta omogućava izbjegavanje ovog problema, a osim tog prednost ovog načina je mogućnost korištenja svih standardnih Linux alata i biblioteka bez potreba za specifičnim sučeljima za komunikaciju između aplikacija (engl. *Application Programming Interface* - API) u stvarnom vremenu [22].

3. IZGRADNJA LINUXA ZA RASPBERRY PI 3 MODEL B POMOĆU BUILDROOT ALATA

Tema ovog diplomskog rada temelji se na izgradnji Linux operacijskog sustava za Raspberry Pi 3 Model B pomoću Buildroota. Buildroot alat omogućava jednostavniju izgradnju Linux operacijskog sustava za odgovarajuće arhitekture. U daljnjem dijelu ovog poglavlja detaljnije su opisani Raspberry Pi 3 Model B, Buildroot alat i koraci prilikom izgradnje Linux operacijskog sustava.

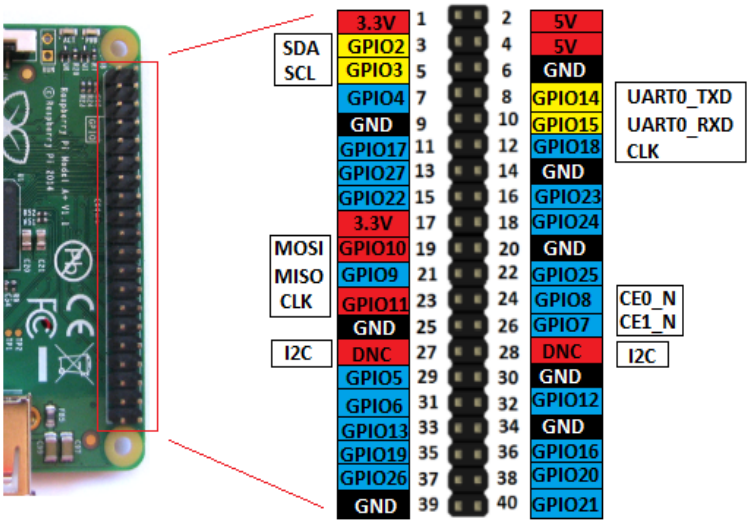
3.1. Raspberry Pi

Raspberry Pi predstavlja seriju jednokartičnih računala koju je izradila Raspberry Pi zaklada u Velikoj Britaniji koja je osnovana 2009. godine, te za cilj ima educiranje ljudi u računalstvu. Do ožujka 2017. godine prodano je preko 12.5 milijuna jedinica, što Raspberry Pi čini trećim najprodavanijim „računalom opće namjene“, ta brojka trenutno iznosi preko 19 milijuna prodanih računala. Godine 2009. mali tim na laboratoriju za računala (engl. *Computer Laboratory*) na Sveučilištu u Cambridgeu shvatili su da je zainteresiranost za računalne znanosti u opadanju, te su kao rješenje problema smatrali jednostavan i jeftin pristup računalu. Širom svijeta ljudi koriste Raspberry Pi kako bi naučili određene vještine programiranja, kako izraditi različite hardverske projekte, izrada kućne automatizacije i slično. Raspberry Pi je vrlo jeftino računalo koje pokreće Linux operacijski sustav, ali također nudi skup ulazno/izlaznih jedinica opće namjene (engl. *General-Purpose Input/Output* - GPIO) koje omogućavaju spajanje različitih senzora ili aktuatora [11]. Raspberry Pi zaklada promovirala je nekoliko različitih modela i verzija Raspberry Pi računala. Prilikom izrade ovog diplomskog rada koristio se Raspberry Pi 3 Model B prikazan na slici 3.1.



Sl. 3.1. *Raspberry Pi 3 Model B [11].*

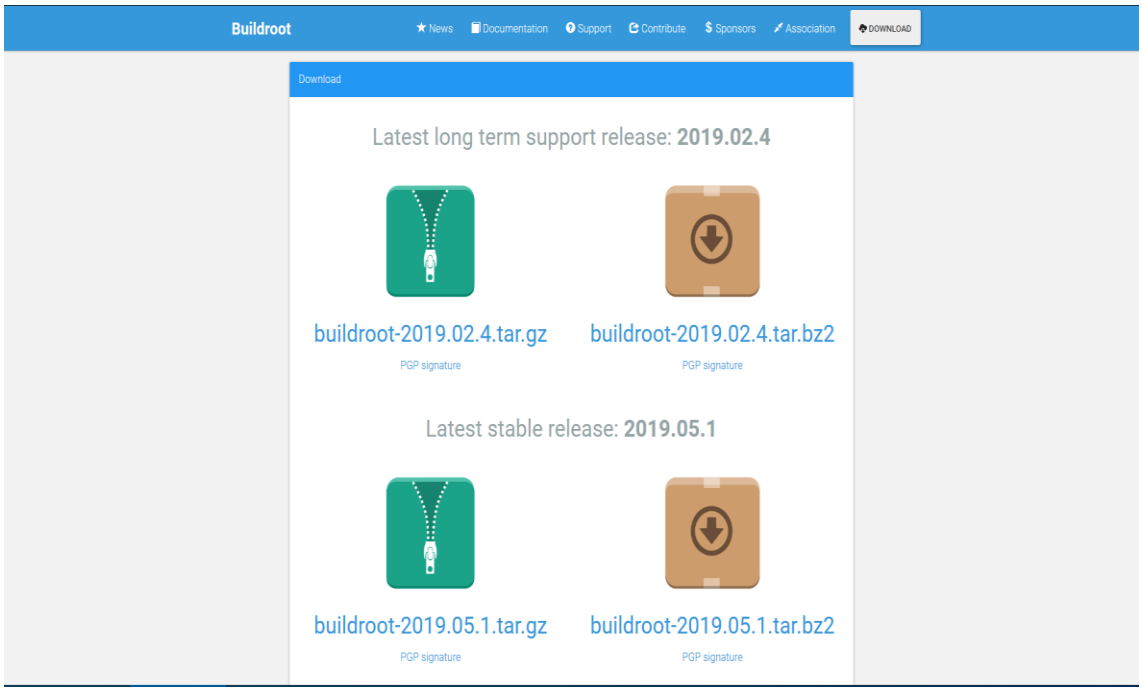
Kada se govori o specifikacijama Raspberry Pi 3 Model B modela, vidljiva je velika razlika u odnosu na njegovu prethodnu verziju. Raspberry Pi 3 Model B sadrži Broadcom BCM2837 sustav na čipu. Broadcom BCM2837 je četverojezgreni procesor snage 1.2 GHz s Cortex-A53 procesorom. Broadcom BCM2837 je 64-bitni procesor, ali glavna prednost ovog procesora je njegova efikasnost i daleko veća razlika u snazi u odnosu na njegovu prethodnu inačicu. Također kada je riječ o grafičkoj procesorskoj jedinici (engl. G*raphics Processing Unit* - GPU) Raspberry Pi 3 Model B sadrži Broadcom VideoCore IV brzine 400 MHz koja je dizajnirana za isporučivanje 1080p videozapisa. Kada je riječ o količini radne memorije Raspberry Pi 3 Model B sadrži 1GB radne memorije. Jedna od glavnih značajki Raspberry Pi-a je niz ulazno/izlaznih jedinica opće namjene koji se nalaze duž gornjeg ruba ploče kao što je vidljivo na slici 3.2.. Na svim Raspberry Pi pločama nalazi se 40 ulazno/izlaznih jedinica opće namjene, gdje se bilo koji pin može koristiti kao ulazni ili izlazni pin opće namjene što je moguće softverski podešavati. GPIO je standardni tip pina koji se može koristiti za uključivanje i isključivanje uređaja, npr. svjetlosnih dioda. GPIO pinovi mogu se koristiti s različitim alternativnim funkcijama. Na slici 3.2. prikazani su pinovi na kojima su dostupne alternativne funkcije poput I2C (engl. *Inter-Intergrated Circuit*), UART (engl. *Universal Asynchronous Reciver/Transmitter*) i SPI (engl. *Serial Peripheral Interface*) koje omogućavaju korisnicima veću fleksibilnost prilikom priključivanja dodatnog hardvera. UART pinovi (prijenos GPIO14, primanje GPIO15) omogućavaju serijsku komunikaciju s Raspberry Pi [26].



Sl. 3.2. *Ulazno/Izlazne jedinice opće namjene Raspberry Pi 3 Model B [9].*

3.2. Buildroot

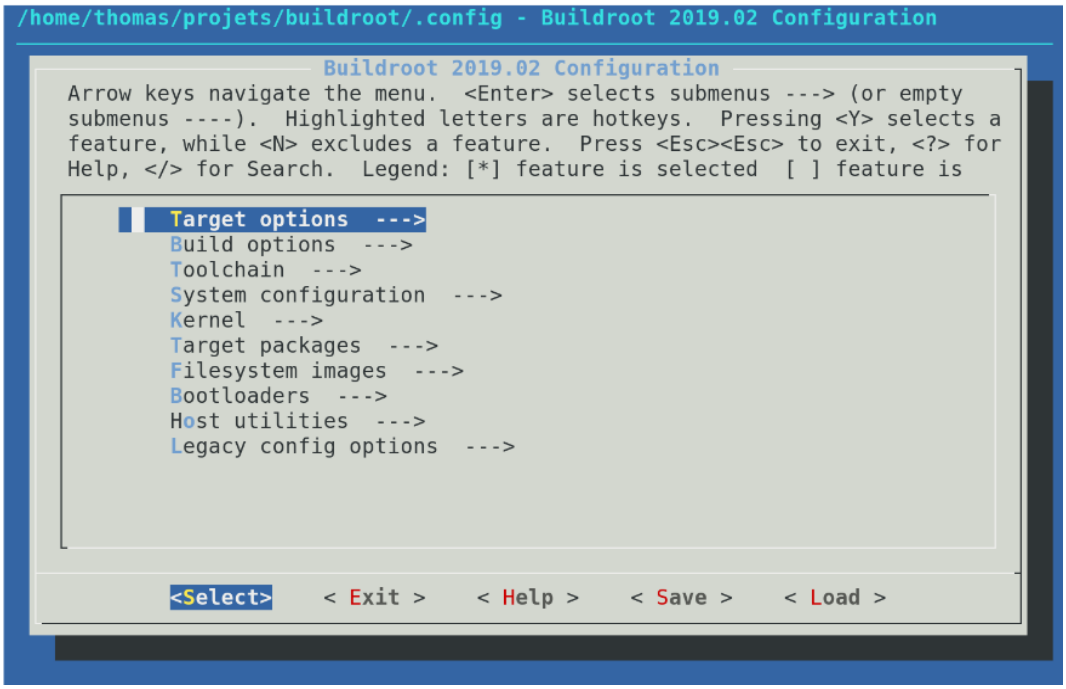
Buildroot je alat koji pojednostavljuje i automatizira proces izgradnje cjelovitog Linux operacijskog sustava za ugradbeni sustav, koristeći unakrsnu izgradnju (engl. *cross compilation)*. Pomoću Buildroota moguće je generirati unakrsni prevoditelj (*engl. compiler*), izraditi jezgru koja se pokreće na arhitekturi koja se razlikuje od one na kojoj se razvija (engl. *cross compilation toolchain*), izgraditi korijenski datotečni sustav i izraditi automatski pokretač operacijskog sustava. Buildroot se koristi za bilo koju kombinaciju prethodno navedenih opcija npr. moguće je upotrebljavati postojeći alat za unakrsnu izgradnju i izgraditi samo korijenski datotečni sustav pomoću Buildroota. Ugradbeni sustavi često koriste procesore koji nisu x86 procesori koji se nalaze u osobnim računalima. Postoje različiti tipovi procesora koje podržava Buildroot kao što su npr. PowerPC procesor, MIPS procesor, ARM procesor i slično. Za preuzimanje Buildroot alata koristi se repozitorij koji je prikazan na slici 3.3. Preporučuje se korištenje najnovije inačice Buildroota budući da se Buildroot brzo razvija te se često dodaju nove značajke.



Sl. 3.3. *Prikaz repozitorija za preuzimanje najnovije inačice Buildroota u trenutku pisanja teksta [14].*

Buildroot alat dolazi sa zadanim konfiguracijama za nekoliko dostupnih razvojnih ploča. Za podešavanje konfiguracije koristi se naredba „`make menuconfig`“, gdje se nakon

pokretanja naredbe prikazuje sučelje kao na slici 3.4 u kojem je moguće odabrati ciljanu arhitekturu, te ostale parametre koje želimo postaviti kako bi izgradili vlastiti Linux operacijski sustav [12].



Sl. 3.4. *Prikaz menuconfig prozora nakon pokretanja make menuconfig naredbe [13].*

Osim gore navedenog načina postavljanja postavki Buildroot alata, postoji još jedan način a to je uređivanje „`.config`" konfiguracijske datoteke. „`Config`" konfiguracijska datoteka sadrži vrijednosti za sve opcije (osim za one koje imaju neispunjene ovisnosti). Zadana „`config`" konfiguracijska datoteka bez ikakvog prilagođavanja sadrži 4074 redaka (2019.02 inačica Buildrota), te nije praktična za korisnike koji ju čitaju i mijenjaju. Osim „`.config`" konfiguracijske datoteke postoji i „`defocnfig`" datoteka koja pohranjuje samo vrijednosti opcija za koje su odabrane zadane vrijednosti. U početnim postavkama Buildroota „`defconfig`" datoteka je prazna. Prilikom promjene arhitekture u ARM arhitekturu, „`defconfig`" datoteka postaje samo jedan redak: BR2_arm = y. Osim arhitekture, postavljanjem ostalih postavki unutar Buildroota „`defconfig`" datoteka se popunjava. Za upotrebu „`defconfig`" datoteke nije dovoljno samo kopirati „`defconfig`" datoteku u „`config`" datoteku jer je potrebno proširivati sve zadane opcije. Buildroot omogućava učitavanje „`defconfig`" datoteke pohranjene u direktoriju na sljedeći način: „`make <foo>_defconfig`" te tako prepisuje „`config`" konfiguracijsku datoteku ako postoji. Za

stvaranje „defconfig" datoteke potrebno je pokrenuti sljedeću naredbu: *„make savedefconfig"*. Za učitavanje „defconfig" datoteke u Buildroot potrebno ju je premjestiti u direktorij konfiguracije (engl. *configs*). Buildroot dolazi s brojnim postojećim „defconfig" datotekama za razne hardverske platforme kao što su: Raspberry Pi, BeagleBone Black, CubieBoard, MicroChip ploče, te platforme koje se emuliraju pomoću QEMU emulatora [13].

Svaki Linux ugradbeni sustav koji je izgrađen pomoću Buildroot alata sastoji se od nekoliko osnovnih direktorija a to su [13]:

- `makefile` – datoteka koja upravlja konfiguracijom sustava,
- `config.in` – konfiguracija najviše razine, sadrži glavne opcije te mnoge druge `config.in` datoteke,
- arhitektura (engl. *architectures*) – `config.in*` datoteke koje definiraju tipove arhitektura (vrste procesora, ABI, itd.). Postoje sljedeći tipovi arhitektura: `Config.in`, `Config.in.arm`, `Config.in.x86`, `Config.in.microblaze` i slično,
- `alat` (engl. t*oolchain*) – odgovarajući paketi za generiranje ili korištenje alata. Prilikom odabira alata postoje dvije mogućnosti: Buildroot alat (engl. *Toolchain - Buildroot*) ili vanjski alat (engl. *toolchain - external*),
- `sistem` (engl. s*ystem*) – sadrži dvije datoteke, gdje se u jednoj nalazi značajke koje se koriste u cijelom sustavu kao što je inicijalizacija sustava (engl. *init system*), dok se u drugoj nalazi kostur sustava (engl. *the rootfs skeleton*),
- `linux` – sadrži *„linux.mk"* što predstavlja jezgru Linux paketa,
- `paketi` (engl. *packages*) – sadrži sve pakete korisničkog sustava. Prilikom izgrade ovog diplomskog rada korišten je BusyBox paket,
- `fs` – predstavlja logiku za generiranje slika datotečnog sustava u različitim formatima. Tipovi formata su: `ext2`, `ext4`, `tar`, `ubifs`, `cpio` i slično,
- `boot` – sadrži pakete automatskog pokretača operacijskog sustava. Tipovi automatskog pokretača su: `uboot`, `barebox`, `grub2`, `syslinux` i slično,
- `konfiguracijske datoteke` – sadrži zadane konfiguracijske datoteke za razne platforme,

- `ploča` (engl. *board*) – sadrži datoteke specifične za ploču (konfiguracijske datoteke jezgre, zakrpe jezgre i slično),
- `podrška` (engl. *support*) – sadrži uslužne programe (`kconfig` kod, zakrpe za `libtool`, pomoćne programe za preuzimanje i drugo),
- `dokumentacija` (engl. *documentation*) – sadrži dokumentaciju Buildroot alata, koja je unaprijed dostupna na mreži,
- `izlaz` (engl. *output*) – predstavlja globalni izlazni direktorij koji se sastoji od sljedećih datoteka:
    - `build` – predstavlja datoteku u kojoj se vrši izrada svakog paketa,
    - `host` – sadrži alate izgrađene za *`host`* (unakrsni prevoditelj i slično), te sadrži *`sysroot`*,
    - `staging` – predstavlja simboličnu vezu na *`sysroot`*,
    - `target` – koristi se za stvaranje konačnih slika korijenskog datotečnog sustava. Buildroot se ne pokreće kao `root`, sve datoteke su u vlasništvu korisnika koji pokreće Buildroot,
    - `images` – sadrži završne slike, sliku jezgre, sliku za pokretanje sustava, slike korijenskog datotečnog sustava,
    - `graphs` – predstavlja vizualizaciju Buildroot operacija, odnosno ovisnosti između paketa, vrijeme za izgradnju različitih paketa,
    - `legal`-info – sadrži pravne podatke, licence za sve pakete i njihov izvorni kod.

3.3. Izgradnja LinuxOS za Raspberry Pi 3 Model B

Kako je poznato da su zadane postavke Linux jezgre lako podesive napredni korisnici imaju mogućnost promijeniti zadane postavke, te ih prilagoditi vlastitim potrebama kao što je omogućavanje novog ili eksperimentalnog mrežnog protokola ili omogućavanje podrške za novi hardver. Za izgradnju Linuxa za Raspberry Pi 3 Model B koristi se osobno računalo koje koristi Ubuntu operacijski sustav, te je zbog različitih arhitektura na kojima se pokreće i na kojoj se razvija Linux potrebno instalirati određene pakete i alate. Nizom sljedećih naredbi omogućeni su svi potrebni paketi i alati:

```
„sudo apt-get install gcc-arm-linux-gnueabihf“
```

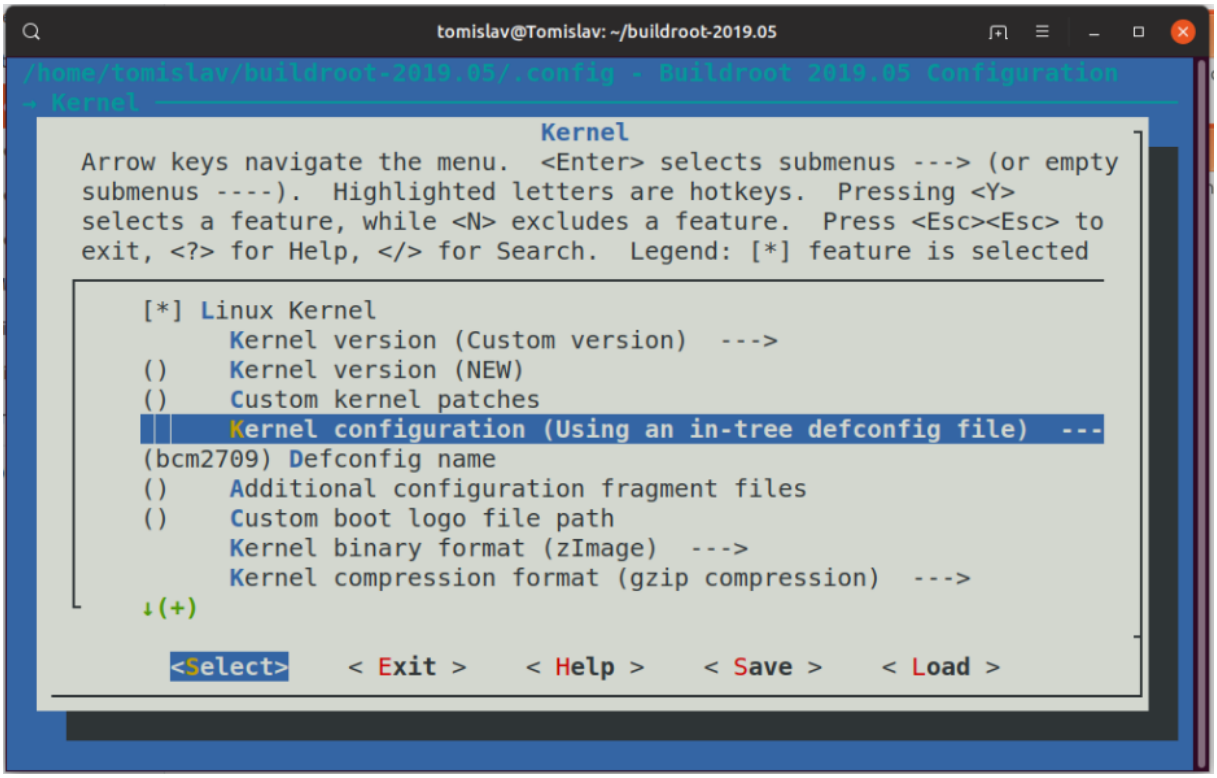
„make git-core ncurses-dev gcc-arm-linux-gnueabihf“

„git clone https://github.com/raspberrypi/tools“

Zadane postavke Linux operacijskog sustava podešavaju se unutar *„menuconfig“* sučelja. Osim ovog načina podešavanja zadanih postavki, postoji još jedan način a to je da ručno izmijenimo datoteku `.config`, ali ovakav način može predstavljati problem kod novih korisnika. Za korištenje *„menuconfig“* sučelja potrebno je pravilno sastaviti *„ncurses“* zaglavlja, zbog čega je potrebno pokrenuti naredbu:

„sudo apt-get install libncurses5-dev“

Nakon instaliranja potrebnih paketa i alata za izradu Linuxa za Raspberry Pi 3 Model B, potrebno je unutar konfiguracijskog sučelja „menuconfig“ odabrati odgovarajuću Linux jezgru. Na slici 3.5. može se vidjeti izgled konfiguracijskog sučelja „menuconfig“ za odabir Linux jezgre.

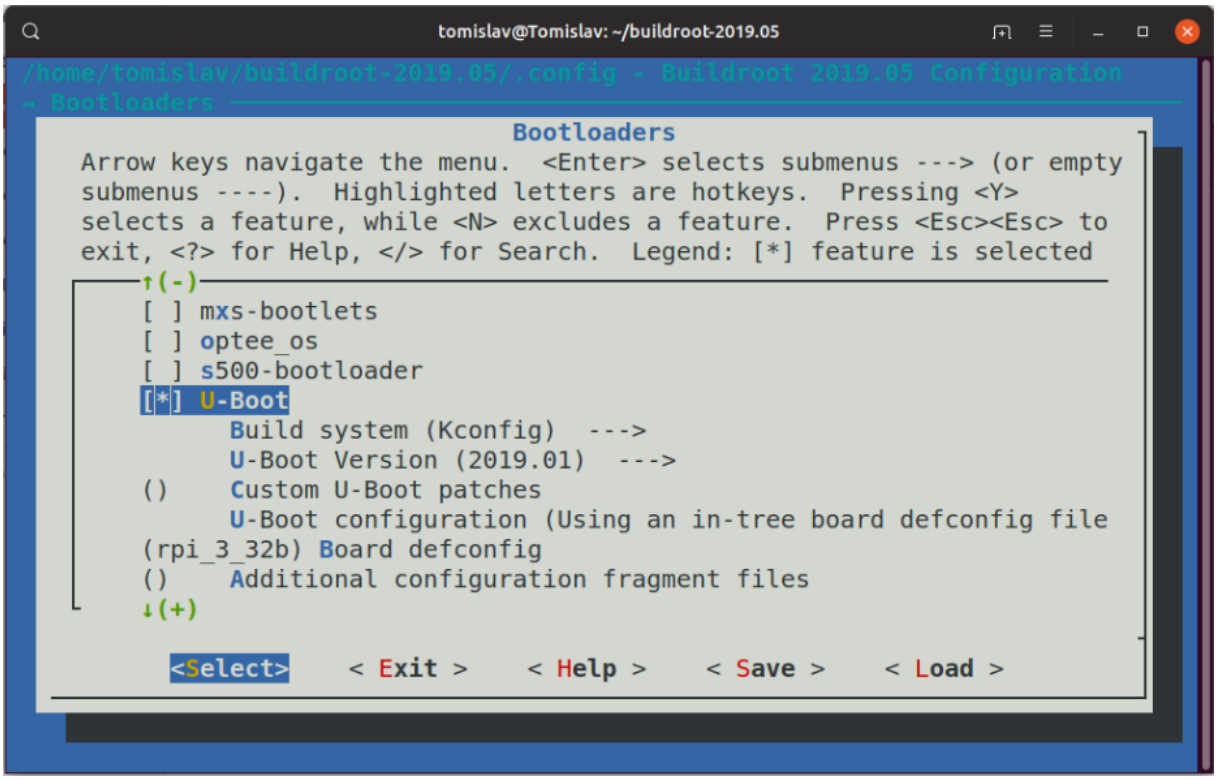


Sl. 3.5. *Prikaz konfiguracijskog sučelja „menuconfig“ za odabir Linux jezgre.*

Izgradnja Linux operacijskog sustava za Raspberry Pi 3 Model B izvršava se unutar konfiguracijskog sučelja „`menuconfig`". Prolaskom kroz različite izbornike unutar „`menuconfig`" sučelja konfigurirani su odgovarajući parametri Buildroot alata za korištenje operacijskog sustava na Raspberry Pi 3 Model B razvojnoj ploči. Na slici 3.6. prikazan je automatski pokretač sustava kao jedan od parametara koji su promijenjeni prilikom postavljanja.

Ostali konfigurirani parametri su sljedeću:

- ciljana arhitektura (engl. *target arhitecture*) – ARM (little endian),
- inačica ciljane arhitekture (engl. *target architecture variant*) – Cortex – A53,
- ciljani ABI (engl. *target ABI*) – EABIhf,
- opcije izrade (engl. *build options*)– zadržane zadane vrijednosti,
- alat (engl. *toolchain*):
    - vrsta alata (engl. *toolchain type*): Buildroot alat,
    - c knjižnica (engl. *c library*) : uClibc-ng,
    - inačica prevoditelja (engl. *gcc compiler version*): gcc 7.x,
- sistemska konfiguracija (engl. *system configuration*):
    - ime sustava (engl. *system hostname*): buildroot,
    - baner sustava (engl. *system banner*): Welcome to Buildroot,
- linux jezgra (engl. *linux kernel*):
    - inačica jezgre (engl. k*ernel version*): Linux 4.14.50,
    - format jezgre (engl. *kernel binary format*): zImage,
    - ime defconfig datoteke (engl. *defconfig name*): bcm2710,
- ciljani paket (engl. *target packages*):
    - busybox,
    - mrežne aplikacije (engl. *networking applications*): Open SSH -> Yes,
    - aplikacije: Open SSH -> Yes,
- slike datotečnog sustava (engl. *filesystem images*),
    - ext2/3/4 root datotečni sustav: ext4, veličina 2048MB,
- automatski pokretač sustava (engl. *bootloader*):
    - uBoot,
- alat za uređivanje teksta (engl. *text editor*):
    - nano alat za uređivanje teksta



Sl. 3.6. *Prikaz odabira automatskog pokretača sustava unutar Buildroot alata.*

Kada su svi koraci odrađeni prema uputama pokreće se naredba za izgradnju Linuxa:

`„make ARCH=arm -j4“`

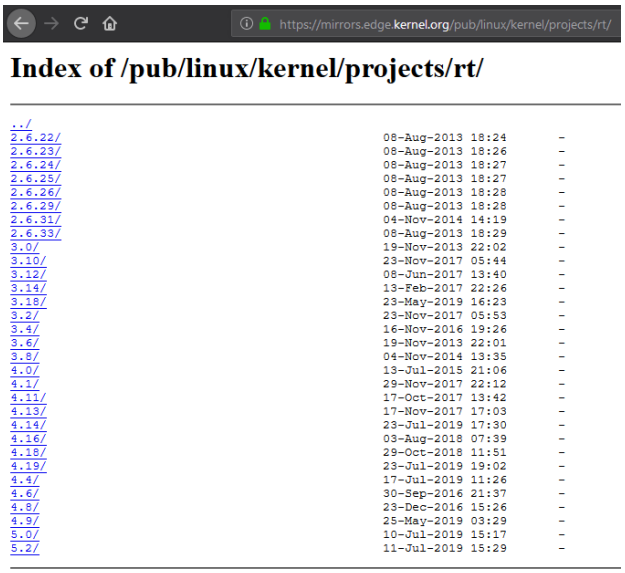
Za bržu izradu Linuxa, ovisno o procesoru moguće je postaviti dodatni parametar *„-j4“* pod pretpostavkom da procesor sadrži četiri jezgre. Na kraju procesa izgradnje Linux operacijskog sustava, Linux slika `zImage`, datotečni sustav `rootfs.ext2` i automatski pokretač sustava `uboot.bin` koji se nalaze u direktoriju *„/home/buildroot-2019.05.1/output/images/“*. Nakon kopiranja odgovarajućih datoteka na mikro SD karticu te pokretanja Linuxa na Raspberry Pi 3 moguće je pomoću naredbe `„uname -a“` vidljivo je o kojoj je inačici riječ kao što je prikazano na slici 3.7.

```
pi@raspberrypi:~ $ uname -a
Linux raspberrypi 4.14.50-v7+ #1122 SMP Tue Jun 18 12:26:26 BST 2019 armv71 GNU/Linux
pi@raspberrypi:~$
```

Sl. 3.7. *Prikaz inačice Linux jezgre pomoću naredbe „uname -a“.*

3.4. Izgradnja LinuxRTOS za Raspberry Pi 3 Model B razvojnu ploču

Za izgradnju LinuxRTOS potrebni su identični alati i paketi kao što su potrebni i za izgradnju standardnog Linux operacijskog sustava za Raspberry Pi 3 Model B. Prilikom izrade LinuxRTOS ponavlja se većina koraka iz prethodnog poglavlja kao što je preuzimanje Linux jezgre s gore navedenog repozitorija, instaliranje potrebnih alata, te postavljanje konfiguracijske jezgre kako bi odgovarala Raspberry Pi 3 Model B ploči. Razlika prilikom izrade LinuxRTOS i standardnog Linux operacijskog sustava je u tome što za izradu LinuxRTOS najprije na standardnu jezgru primjenjujemo odgovarajuću zakrpu koja omogućava rad u stvarnom vremenu, a nakon toga se pokreće postupak konfiguriranja i prevođenja. Za izradu LinuxRTOS potrebno je preuzeti odgovarajuću zakrpu s repozitorija koji je vidljiv na slici 3.8.



Sl. 3.8. *Prikaz repozitorija za preuzimanje zakrpe za rad u stvarnom vremenu [16].*

Na kraju nakon preuzimanja odgovarajuće Linux jezgre i odgovarajuće zakrpe za rad u stvarnom vremenu, potrebno je pomoću naredbe:

```
„make menuconfig“
```

pokrenuti konfiguracijsko sučelje u kojem je potrebno unutar općenitih postavki (engl. *General Setup/Preemption Model*) odabrati (engl. *Fully preemptible kernel*). Kao i kod izrade LinuxOS, konfigurirani su parametri potrebni za izgradnju LinuxRTOS što je vidljivo u prethodnom

poglavlju (3.3). Nakon postavljanja svih postavki potrebno je snimiti konfiguracijsku datoteku te pokrenuti izgradnju Linux jezgre za rad u stvarnom vremenu pomoću naredbe:

„`make ARCH=arm -j4`“

Na kraju procesa izgradnje LinuxRTOS u direktoriju *„/home/buildroot-2019.05.1/output/images/“* nalaze se iste datoteke kao i u prethodnom poglavlju, samo u ovom slučaju Linux jezgra je *preemptible* kao što se može vidjeti na slici 3.9. nakon što se pokrene naredba „`uname-a`“.

```
pi@raspberrypi:~ $ uname –a
Linux realtimepi 4.14.52-rt34-v7 #2 SMP PREEMPT RT Tue Jun 10 14:28:22 BST 2019 armv71
pi@raspberrypi:~ $
```

Sl. 3.9. *Prikaz inačice Linux jezgre pomoću naredbe „uname -a“.*

3.5. Izgradnja Linuxa za QEMU emulator

Razvoj softvera za ugradbene sustave tradicionalno se razvija na osobnom računalu uz pomoću unakrsnog prevoditelja, gdje se nalazi instalirana Linux jezgra i pripadajući datotečni sustav. Alternativni pristup je korištenje emulatora kao što je QEMU kako bi se započeo razvojni proces bez temeljnog hardvera. QEMU emulator (engl. *Quick emulator*) je besplatni emulator otvorenog koda koji izvodi virtualizaciju hardvera. Postoje dva načina rada QEMU emulatora, a to su [27]:

- potpuna emulacija sustava gdje QEMU emulator oponaša cijeli sustav uključujući jedan ili više procesora i razne periferne jedinice. Može se koristiti za pokretanje različitih operacijskih sustava bez ponovnog pokretanja računala,
- emulacija korisničkog načina gdje QEMU emulator može pokrenuti procese sastavljene za jedan CPU na drugom CPU.

Iako se ovaj pristup koristi u ranim fazama razvoja softvera, on može u potpunosti zamijeniti tradicionalni pristup. Korištenje emulatora procesora ima i akademsku vrijednost što korisnicima omogućuje upoznavanje s LinuxRTOS i tehnikama razvoja softvera za ugradbene sustave. Prilikom izrade ovog diplomskog rada QEMU emulator koristio se u početnim fazama razvoja Linux operacijskog sustava. Kako se svaki korak izgradnje vlastite Linux operacijskog sustava izvodi preko naredbenog retka, nakon preuzimanja odgovarajuće inačice Buildroota

potrebno je raspakirati mapu *„buildroot-2019.05.1.tar.bz2“* te promijeniti trenutnu putanju u *„/home/buildroot-2019.05.1/“*. Za pokretanje konfiguracijskog sučelja Buildroota koristi se naredba:

„make menuconfig“

u kojem postoji mogućnost podešavanja cjelokupnog Linux operacijskog sustava. Prije pokretanja konfiguracijskog sučelja Linux operacijskog sustava potrebno je pomoću naredbe:

„make qemu_arm_versatile_defconfig“

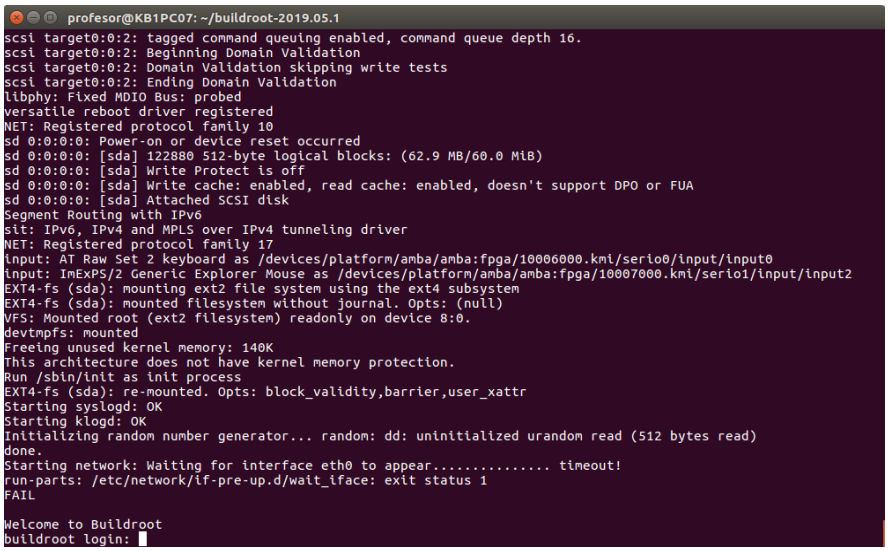
postaviti konfiguracijsku jezgru kako bi odgovarala ARM Versatile boards predlošku. ARM Versatile Boards predstavlja jednu od razvojnih platformi koji podržavaju Linux i QEMU. Nakon odabira svih nužnih paketa, potrebno je spremiti konfiguracijsku datoteku i pokrenuti izgradnju Linux jezgre pomoću naredbe *„make“*. Na kraju procesa izgradnje Linux jezgre, Linux slika `zImage` i njegov datotečni sustav `rootfs.ext2` nalaze se u direktoriju *„/home/buildroot-2019.05.1/output/images/“*. Kako bi se gotova Linux jezgra uspješno pokrenula na osobnom računalu potrebno je pomoću naredbe:

„sudo apt-get install qemu“

instalirati QEMU emulator. Posljednji korak je primjena QEMU emulatora, te zbog toga nije potrebno koristiti niti jedan od automatskih pokretača operacijskih sustava kao što je npr. `U-Boot`, nego je moguće odmah nakon završetka izgradnje jezgre pokrenut naredbu:

„qemu-system-arm -M versatilepb -kernel output/images/zImage -dtb output/images/versatile-pb.dtb -drive file=output/images/rootfs.ext2,if=scsi -append "root=/dev/sda console=ttyAMA0,115200" -nographic“.

Nakon završetka svih navedenih koraka i ako se sve izvršilo bez pogreške potrebno je unutar naredbenog retka dobiti prikaz kao što je na slici 3.10.



Sl. 3.10. *Prikaz Linux operacijskog sustava na QEMU emulatoru izgrađene pomoću Buildroot alata.*

4. USPOREDBA REZULTATA MJERENJA KOD LINUX OPERACIJSKIH SUSTAVA

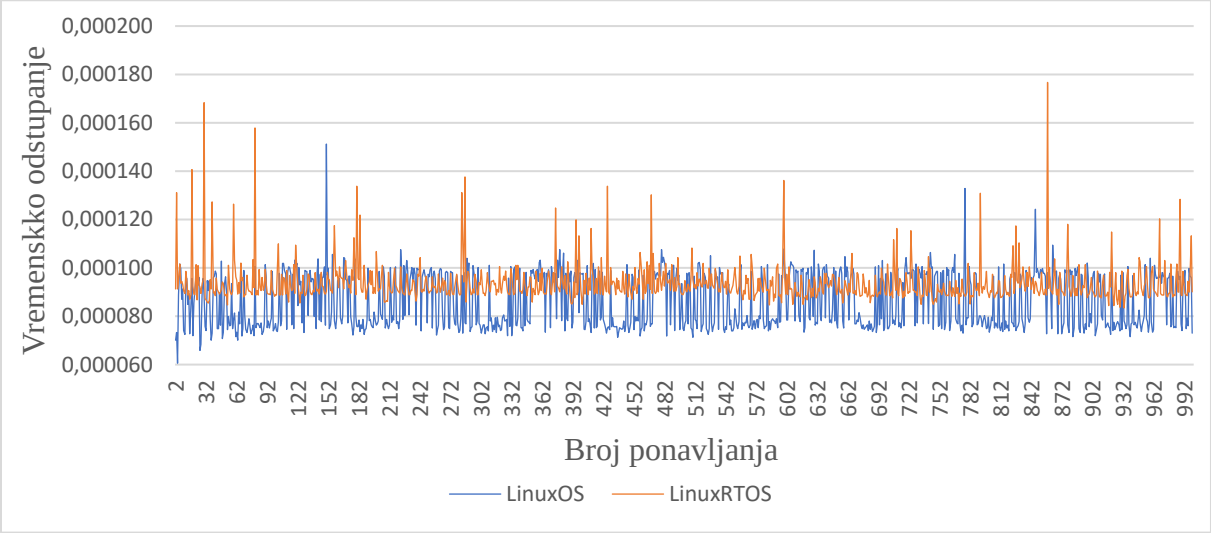
Tema ovog diplomskog rada je pronaći način kako analizirati performanse LinuxOS i LinuxRTOS. Prilikom izrade diplomskog rada napravljeno je šest različitih mjerenja na oba operacijska sustava. Mjerenjima se nastojao pokazati utjecaj zakrpe na rad Linux jezgre za potrebe rada u stvarnom vremenu. Mjerenja koja su provedena na LinuxOS i LinuxRTOS su sljedeća:

- mjerenje mirovanja u sekundama – ovo mjerenje napravljeno je pomoću funkcije *Sleep(),*
- mjerenje mirovanja u mikrosekundama – ovo mjerenje napravljeno je pomoću funkcije *Usleep()* koja obustavlja izvršavanje trenutne niti sve dok ne protekne broj mikrosekundi u stvarnom vremenu koje su specificirane razdobljem ili se signal dostavlja pozivnom nizu. Vrijeme obustavljanja može trajati duže od traženog jer je vrijednost argumenta zaokružena na cijeli broj ili zbog raspoređivanja druge aktivnosti sustava,
- mjerenje vremena potrebnog za čitanje teksta iz odgovarajuće datoteke – ovo mjerenje izvršeno je pomoću funkcije *Read()* koja čita broj bajtova iz odgovarajuće datoteke,
- mjerenje vremena potrebnog za zapisivanje teksta u odgovarajuću datoteku – ovo mjerenje napravljeno je pomoću funkcije *Write()* koja zapisuje odgovarajući niz znakova iz ranije definiranog međuspremnika u odgovarajuću datoteku,
- cikličko mjerenje – ovo mjerenje napravljeno je pomoću test programa koji pruža jednostavan način za procjenu maksimalne latencije sustava. Cikličko mjerenje (engl. *Cycletest*) u osnovi mjeri koliko je potrebno da se odgovori na prekid, a u slučaju cikličnog testnog programa, prekid se generira pomoću sata (engl. *Timera*).

Za svako od navedenih mjerenja napravljeni su programi u C programskom jeziku koje se pokreću na Raspberry Pi 3 Model B ploči i nalaze se u Prilogu 1. Svako mjerenje na LinuxOS i na LinuxRTOS odrađeno je 1000 puta, osim na cikličkom mjerenju gdje je broj ponavljanja iznosio 1 000 000 kako bi bilo lakše usporediti dobivena mjerenja.

4.1. Mjerenje mirovanja u sekundama

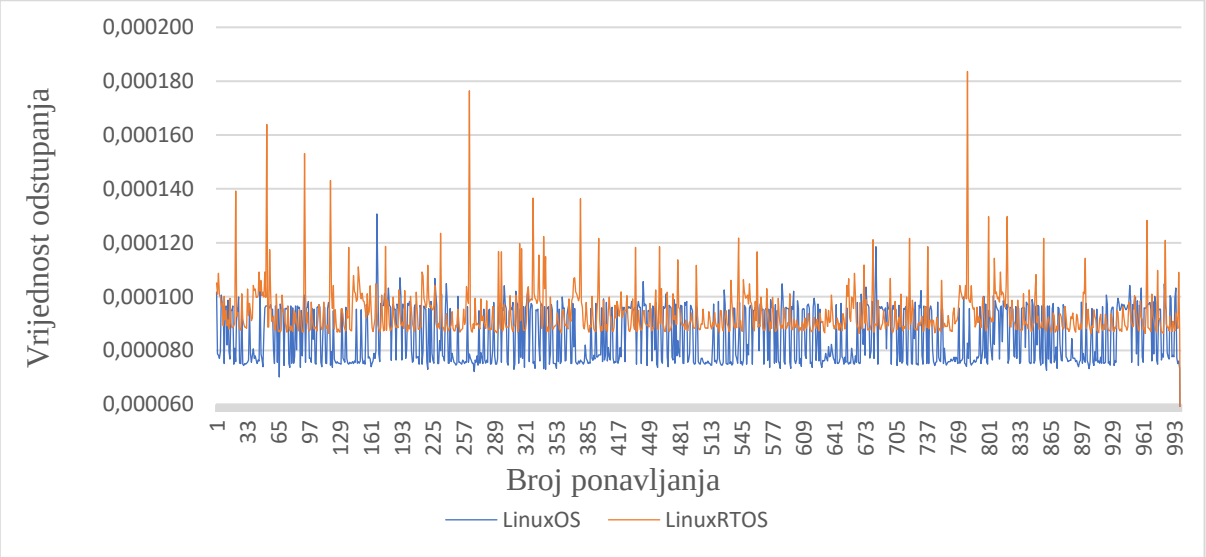
Na slici 4.1. grafički je prikazana usporedba mjerenja odstupanja mirovanja u sekundama od zadane vrijednosti mirovanja LinuxOS i LinuxRTOS pomoću programa napisanog u C programskom jeziku. Zadana vrijednost mirovanja iznosi 5 sekundi. Maksimalno vrijeme potrebno za izvođenje mjerenja mirovanja kod LinuxOS iznosi 5,00015120 sekundi, minimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,00006060 sekundi. Srednja vrijednost vremena potrebnog za izvođenje mjerenja mirovanja kod LinuxOS iznosi 5,00008625 sekundi, a vrijednost standardne devijacije iznosi 11,6 mikrosekundi. Kod LinuxRTOS, maksimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,0001767 sekundi, minimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,0000834 sekundi. Srednja vrijednost vremena potrebnog za izvođenje mjerenja kod LinuxRTOS iznosi 5,000093525 sekundi, a vrijednost standardne devijacije iznosi 8,12 mikrosekundi Narančastom bojom na grafu prikazane su vrijednosti odstupanja od zadane vrijednosti mirovanja za LinuxRTOS, dok su plavom bojom prikazane vrijednosti odstupanja od zadane vrijednosti mirovanja za LinuxOS. Iz slike se može zaključiti da je kašnjenje kod LinuxRTOS u prosjeku nešto veće od kašnjenja LinuxOS, ali zato ima manje rasipanje oko srednje vrijednosti.



Sl. 4.1. *Grafički prikaz usporedbe odstupanja mirovanja u sekundama kod LinuxOS i LinuxRTOS.*

4.2. Mjerenje mirovanja u mikrosekundama

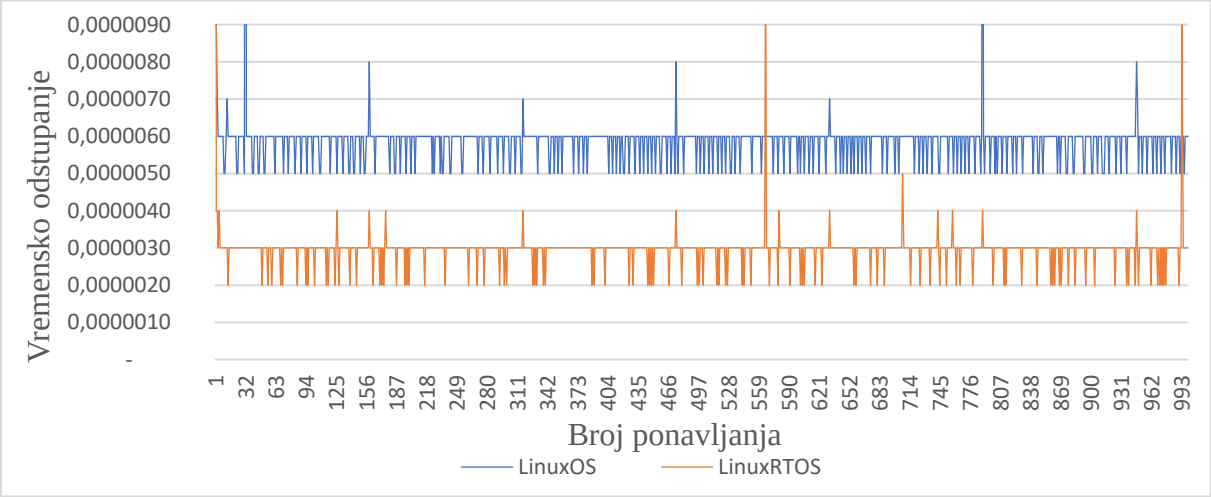
Na slici 4.2. grafički je prikazana usporedba mjerenja odstupanja mirovanja u sekundama od zadane vrijednosti mirovanja LinuxOS i LinuxRTOS pomoću programa napisanog u C programskom jeziku. Zadana vrijednost mirovanja iznosi 5 sekundi. Maksimalno vrijeme potrebno za izvođenje mjerenja mirovanja kod LinuxOS iznosi 5,0001533 sekundi, minimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,0000701 sekundi. Srednja vrijednost vremena potrebnog za izvođenje mjerenja mirovanja kod LinuxOS iznosi 5,00008563 sekundi, a vrijednost standardne devijacije iznosi 10,7 mikrosekundi. Kod LinuxRTOS, maksimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,0001836 sekundi, minimalno vrijeme potrebno za izvođenje mjerenja mirovanja iznosi 5,0000863 sekundi. Srednja vrijednost vremena potrebnog za izvođenje mjerenja kod LinuxRTOS iznosi 5,000093515 sekundi, a vrijednost standardne devijacije iznosi 9,14 mikrosekundi Narančastom bojom na grafu prikazane su vrijednosti odstupanja od zadane vrijednosti mirovanja za LinuxRTOS, dok su plavom bojom prikazane vrijednosti odstupanja od zadane vrijednosti mirovanja za LinuxOS. Iz slike se može zaključiti da je kašnjenje kod LinuxRTOS u prosjeku nešto veće od kašnjenja LinuxOS, ali zato ima manje rasipanje oko srednje vrijednosti.



Sl. 4.2. *Grafički prikaz usporedbe odstupanja mirovanja u mikrosekundama kod LinuxOS i LinuxRTOS.*

4.3. Mjerenje vremena potrebnog za čitanje teksta iz odgovarajuće datoteke

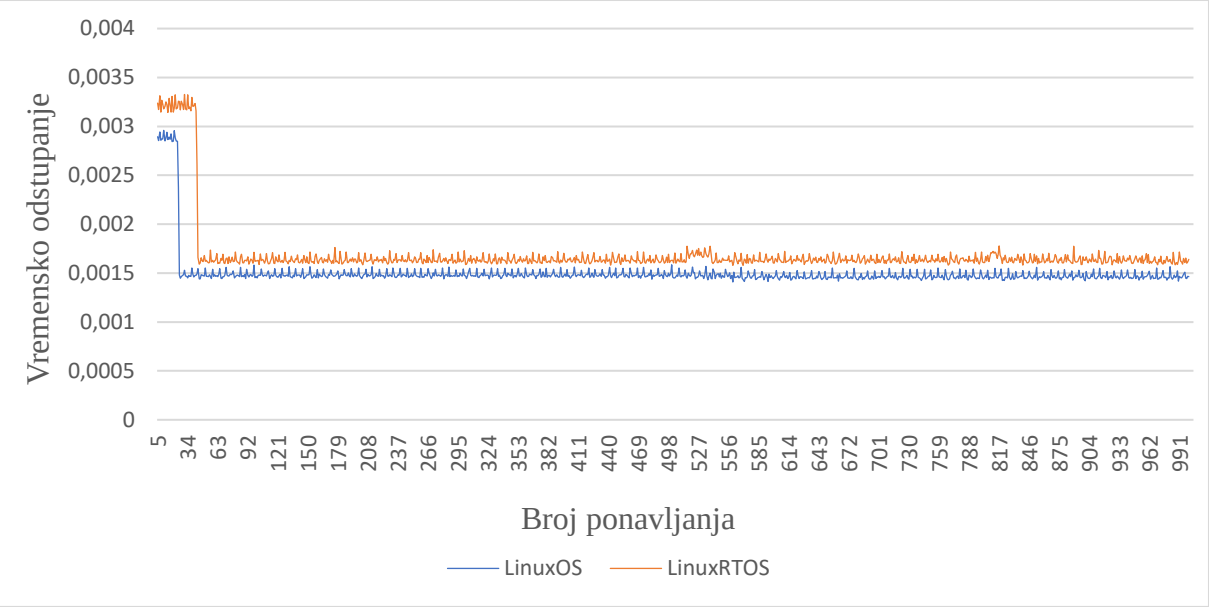
Na slici 4.3. grafički je prikazana usporedba mjerenja vremena potrebnog za čitanje teksta iz odgovarajuće datoteke za LinuxOS i LinuxRTOS pomoću programa napisanog u C programskom jeziku. Maksimalno vrijeme potrebno za čitanje teksta iz odgovarajuće datoteke kod LinuxOS iznosi 0,4512134 sekundi, a minimalna vrijednost iznosi 5 mikrosekundi. Srednje vrijeme potrebno za čitanje teksta iz odgovarajuće datoteke kod LinuxOS iznosi 457,1 mikrosekunda, a vrijednost standardne devijacije iznosi 0,014268434 sekundi. Kod LinuxRTOS, maksimalno vrijeme potrebno za čitanje teksta iz odgovarajuće datoteke iznosi 0,2347201 sekundi, minimalna vrijednost iznosi 2 mikrosekunde. Srednje vrijeme potrebno za čitanje teksta iz odgovarajuće datoteke kod LinuxRTOS iznosi 237,6 mikrosekundi, a vrijednost standardne devijacije iznosi 7422,41 mikrosekunda. Narančastom bojom na grafu prikazana su vremena potrebna za izvršavanje operacije čitanja datoteke kod LinuxRTOS, dok su plavom bojom prikazana vremena potrebna za izvršavanje operacije čitanja datoteke za LinuxOS. Kao što je vidljivo iz slike 4.3. LinuxRTOS ovaj zadatak obavlja značajno brže.



Sl. 4.3. *Grafički prikaz vremena potrebnog za čitanje teksta iz odgovarajuće datoteke kod standardnog LinuxOS i LinuxRTOS.*

4.4. Mjerenje vremena potrebnog za zapisivanje teksta u odgovarajuću datoteku

Na slici 4.4. grafički je prikazana usporedba mjerenja vremena potrebnog za zapisivanje teksta u odgovarajuću datoteku za LinuxOS i LinuxRTOS pomoću programa napisanog u C programskom jeziku. Maksimalno vrijeme potrebno za zapisivanje teksta u odgovarajuću datoteku kod LinuxOS iznosi 3154 mikrosekunde, minimalna vrijednost iznosi 1409 mikrosekunde. Srednje vrijeme potrebno za zapisivanje teksta u odgovarajuću datoteku kod LinuxOS iznosi 1509,7 mikrosekundi, a vrijednost standardne devijacije iznosi 221,7 mikrosekundi. Kod LinuxRTOS, maksimalno vrijeme potrebno za zapisivanje teksta u odgovarajuću datoteku iznosi 3482 mikrosekunde, a minimalna vrijednost iznosi 1579 mikrosekundi. Srednje vrijeme potrebno za zapisivanje teksta u odgovarajuću datoteku kod LinuxRTOS iznosi 1704 mikrosekunde, a vrijednost standardne devijacije iznosi 322,24 sekundi. Narančastom bojom na grafu prikazana su vremena potrebna za izvršavanje operacije zapisivanja teksta u datoteku kod LinuxRTOS, dok su plavom bojom prikazana vremena potrebna za izvršavanje operacije zapisivanja teksta u datoteku za LinuxOS.



Sl. 4.4. *Grafički prikaz vremena potrebnog za zapisivanje teksta u odgovarajuću datoteku kod LinuxOS i LinuxRTOS.*

4.5. Mjerenje maksimalnog kašnjenja jezgri – ciklično mjerenje

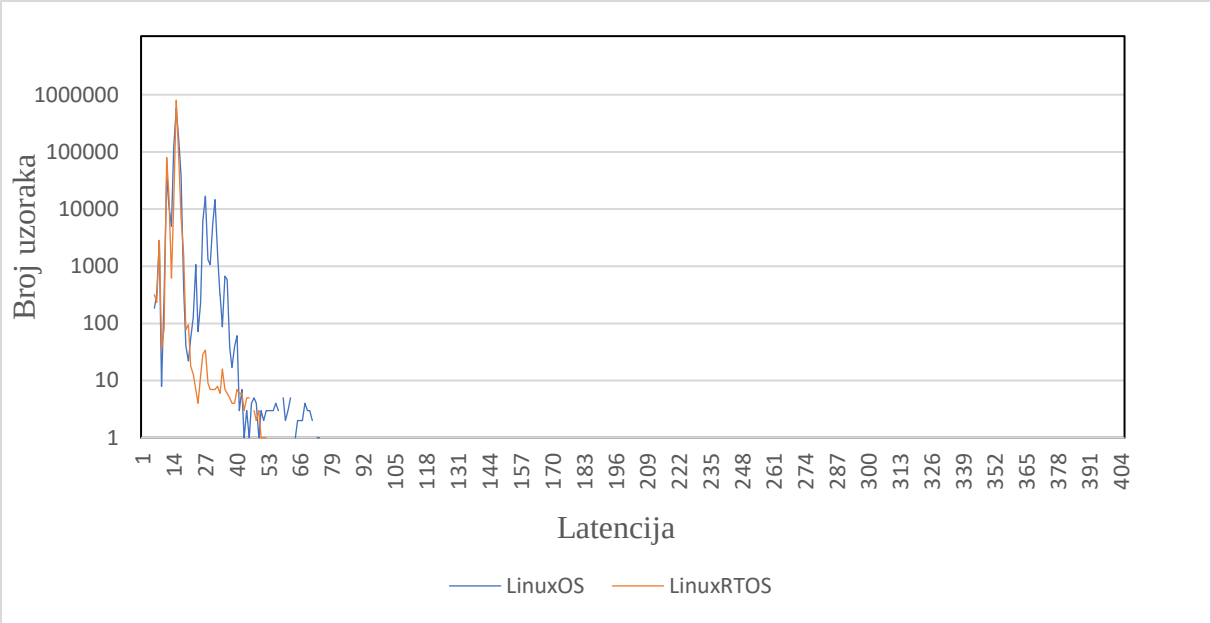
Za razliku od prethodnih mjerenja za mjerenje maksimalnog kašnjenja jezgri potrebno je s odgovarajućeg repozitorija *„rt-tests/rt-tests.git“* preuzeti testove koji se provode u stvarnom vremenu (engl. *Real-Time Test*). Nakon preuzimanja mape s potrebnim testovima za mjerenje smo koristili naredbu koja se izvodi sa sljedećim parametrima :

```
cyclictest -l1000000 -m -S -p90 -i200 -h400 --
histfile=time_cyclictest.txt
```

Prvi parametar naredbe *„l1000000*“ predstavlja broj iteracija koji se koristi prilikom mjerenja. Drugim parametrom *„-m“* zaključavaju se buduće i trenutne raspodjele memorije kako bi se spriječilo ispisivanje. Treći parametar *„S“* predstavlja SMP (engl. *Symmetric multiprocessing*) podršku koja stvara onoliko niti koliko postoji jezgri procesora, te stvorene niti pokreće na istoj jezgri. Četvrti parametar *„-i200“* predstavlja osnovni interval niti, u ovom primjeru iznosi 200 µs. Peti parametar *„-h400“* predstavlja maksimalno kašnjenje na histogramu koje iznosi 400 µs. Zadnji parametar naredbe „*–histfile=time_cyclictest.txt“* predstavlja dokument u koji se pohranjuju dobivene vrijednosti prilikom izvođenja mjerenja.

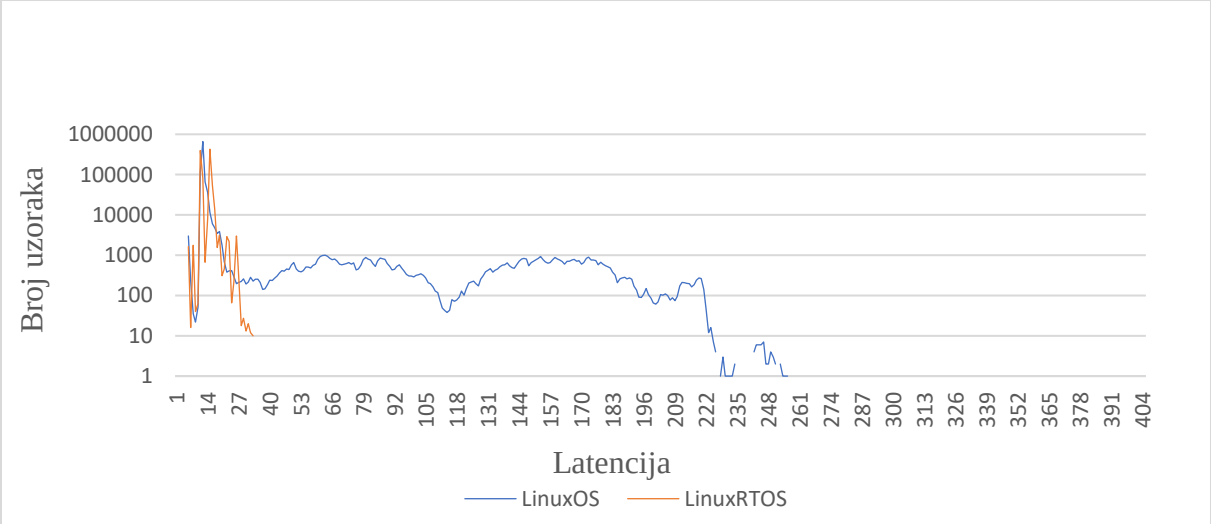
Na slikama (4.5, 4.6, 4.7, 4.8) prikazani su grafički prikazi rezultata mjerenja kod LinuxOS i LinuxRTOS. Na svakom od prikazanih grafova napravljena je usporedba jezgri LinuxOS i LinuxRTOS. Horizontalna os predstavlja vremensko kašnjenje prikazano u mikrosekundama, a vertikalna os predstavlja broj uzoraka latencije.

Na slici 4.5. prikazana je usporedba prve jezgre kod standardnog LinuxOS i LinuxRTOS. U grafu na slici 4.5. vidljivo je da maksimalno kašnjenje prve jezgre kod standardnog LinuxOS iznosi približno 75 µs, dok maksimalno kašnjenje prve jezgre kod LinuxRTOS iznosi približno 53 µs.



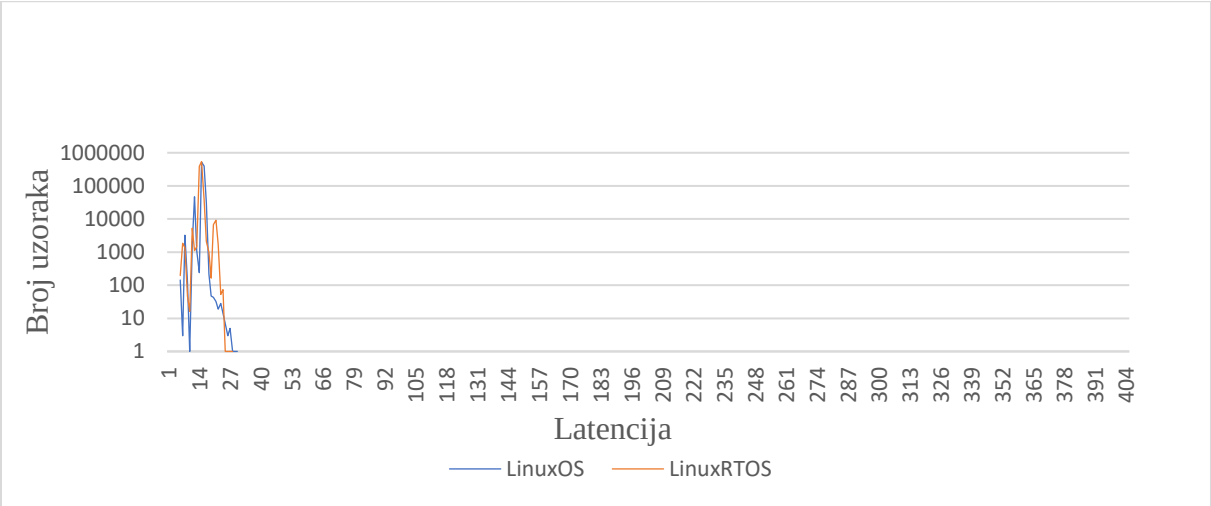
Sl. 4.5. *Grafički prikaz rezultata mjerenja maksimalnog kašnjenja za prvu jezgru kod LinuxOS i LinuxRTOS.*

Na slici 4.6. prikazana je usporedba maksimalnog kašnjenja druge jezgre kod LinuxOS i LinuxRTOS. U grafu na slici 4.6. vidljivo je da maksimalno kašnjenje druge jezgre LinuxOS iznosi približno 255 µs, dok maksimalno kašnjenje druge jezgre LinuxRTOS iznosi približno 38 µs.



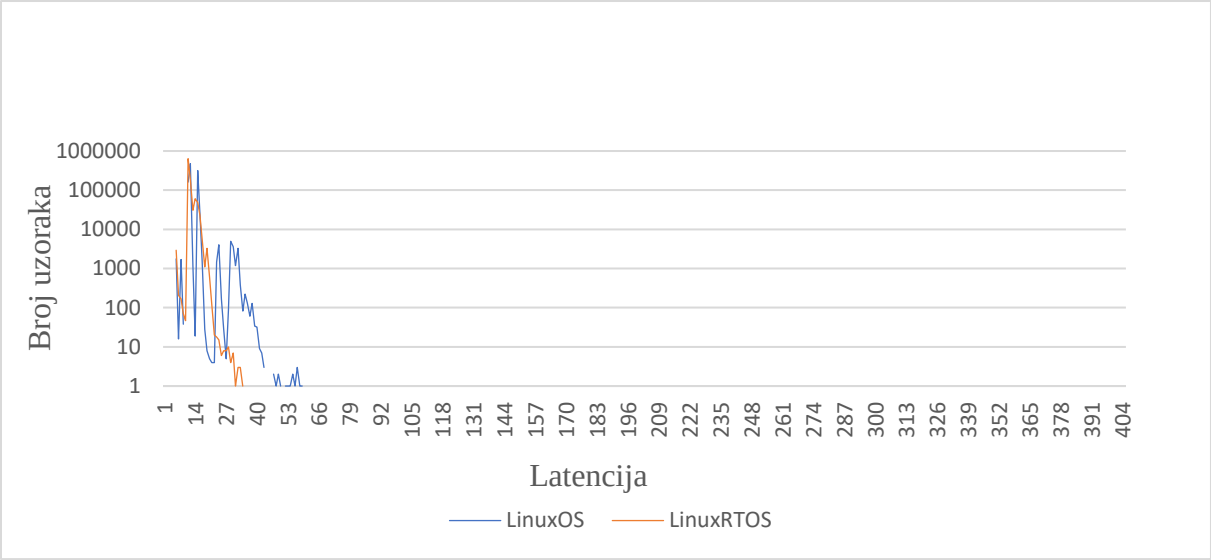
Sl. 4.6. *Grafički prikaz rezultata mjerenja maksimalnog kašnjenja za drugu jezgru kod LinuxOS i LinuxRTOS.*

Na slici 4.7. prikazana je usporedba maksimalnog kašnjenja treće jezgre kod LinuxOS i LinuxRTOS. U grafu na slici 4.7. vidljivo je da maksimalno kašnjenje treće jezgre kod LinuxOS iznosi približno 31 µs, dok maksimalno kašnjenje treće jezgre kod LinuxRTOS iznosi približno 27 µs.



Sl. 4.7. *Grafički prikaz rezultata mjerenja maksimalnog kašnjenja za treću jezgru kod LinuxOS i LinuxRTOS.*

Na slici 4.8. prikazana je usporedba maksimalnog kašnjenja četvrte jezgre kod LinuxOS i LinuxRTOS. U grafu na slici 4.8. vidljivo je da maksimalno kašnjenje četvrte jezgre LinuxOS iznosi približno 58 µs, dok maksimalno kašnjenje četvrte jezgre LinuxRTOS iznosi približno 35 µs. Promatrajući dobivena mjerenja vidljivo je da su maksimalne latencije i srednje vrijednosti latencija za sve jezgre znatno manje kod LinuxRTOS u odnosu na LinuxOS kao što je bilo za očekivati.

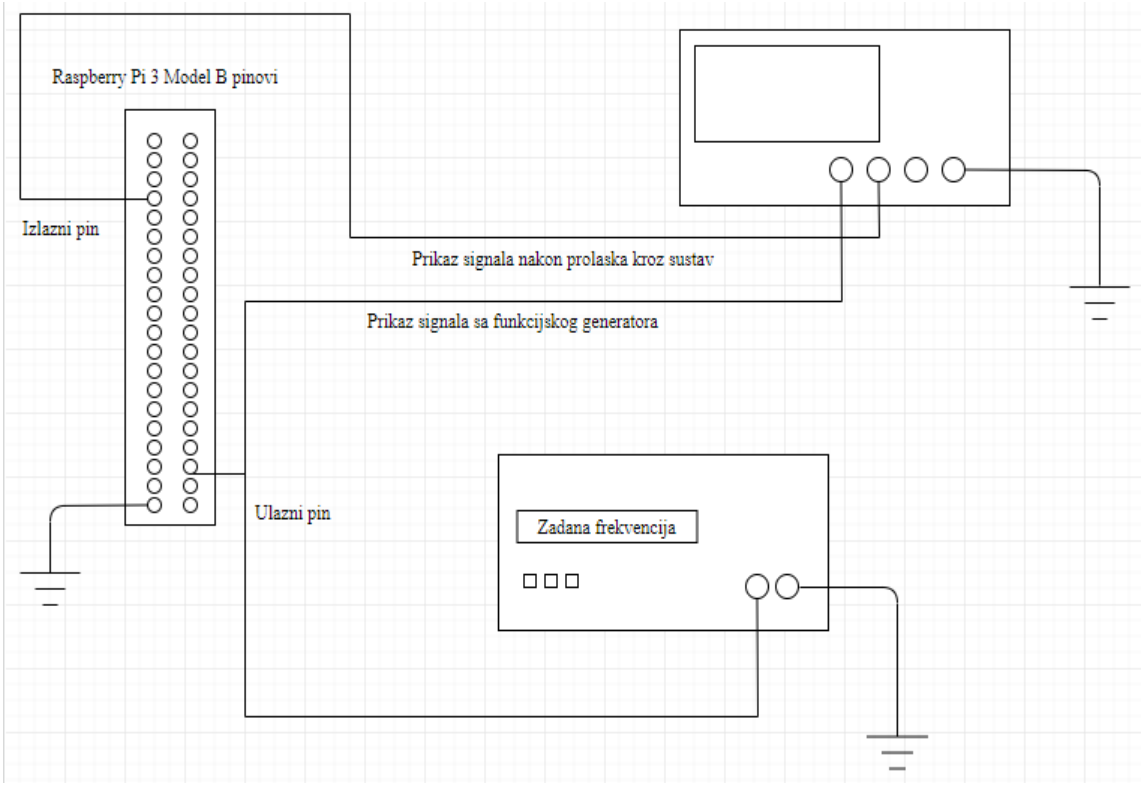


Sl. 4.8. *Grafički prikaz rezultata mjerenja maksimalnog kašnjenja za četvrtu jezgru kod LinuxOS i LinuxRTOS.*

4.6. Mjerenje brzine promjene signala na ulazima/izlazima opće namjene

Prethodna mjerenja temeljena su isključivo na operacijskom sustavu koji se nalazi na Raspberry Pi razvojnoj ploči, dok mjerenje brzine promjene signala na ulazima/izlazima opće namjene ovisi o različitim vanjskim utjecajima, te ispravnosti uređaja koji se koriste tijekom mjerenja. Pomoću ovog mjerenja nastoji se ustanoviti koliko je kašnjenje izlaznog signala u odnosu na ulazni signal i hoće li doći do izobličenja signala s povećanjem frekvencije signala. Mjerenja su obavljena tako da je Raspberry Pi 3 spojen preko svojih ulazno/izlaznih pinova opće namjene s mjernom opremom kao što je prikazano na slici 4.9. Mjerna oprema sastoji se od funkcijskog generatora (RIGOL DF1022A), osciloskopa (RIGOL MSO1104) i odgovarajućih mjernih sondi. Pokus se zasniva na mjerenju razlike između pravokutnog signala koji se šalje s funkcijskog generatora na ulazni pin Raspberry Pi 3 i koji se zatim softverski preusmjerava na odgovarajući izlazni pin. Ulazni i izlazni signal promatraju se na osciloskopu te bi u idealnom slučaju trebali biti identični. Međutim, zbog karakteristika jezgre operacijskog sustava postoje odstupanja između ova dva signala. Prilikom provedbe pokusa sva mjerna oprema kao i Raspberry Pi 3 bili su propisno uzemljeni. Na Raspberry Pi 3 korišten je program koji se nalazi u prilogu 1, a za rad s ulazno/izlaznim pinovima opće namjene koristi se *wiringPi.h* biblioteka [29]. Kao i u prethodnim mjerenjima korištena su dva tipa operacijskog sustava LinuxOS i LinuxRTOS. Mjerenja su izvršena za različite frekvencije od 1 kHz do 900 kHz, gdje su koraci u početku iznosili 10 kHz, a nakon 100 kHz korak se povećavao za 100 kHz. Kašnjenje izlaznog signala u odnosu na ulazni signal kao i frekvencija izlaznog signala

određene su pomoću funkcija dostupnih na osciloskopu. U tablici 4.1. prikazana su dobivena mjerenja.



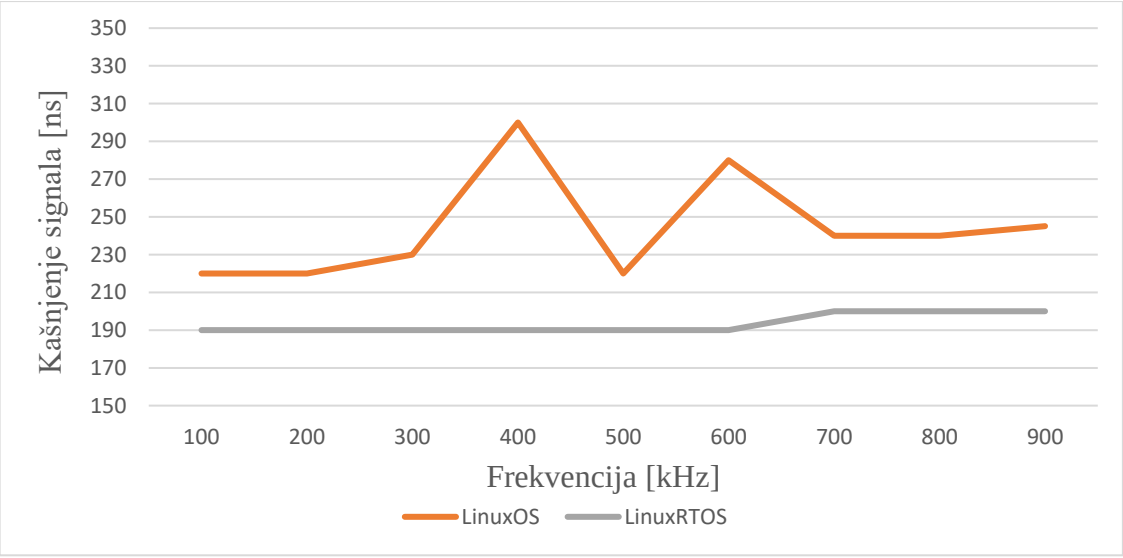
Sl. 4.9. *Shematski prikaz povezivanja Raspberry Pi 3 Model B ploče, funkcijskog generatora i osciloskopa.*

Tablica 4.1. *Prikaz rezultata za LinuxOS i LinuxRTOS*

Mjerenje brzine promjene signala na ulazima/izlazima opće namjene				
Frekvencija funkcijskog generatora	Frekvencija nakon prolaska kroz Raspberry Pi 3 operacijski sustav		Kašnjenje signala u odnosu na signal s funkcijskog generatora	
	LinuxOS	LinuxRTOS	LinuxOS	LinuxRTOS
1 kHz	1 kHz	1 kHz	220 ns	190 ns
10 kHz	10 kHz	10 kHz	220 ns	190 ns
20 kHz	20 kHz	20 kHz	230 ns	190 ns
30 kHz	29.9 kHz	30 kHz	300 ns	190 ns

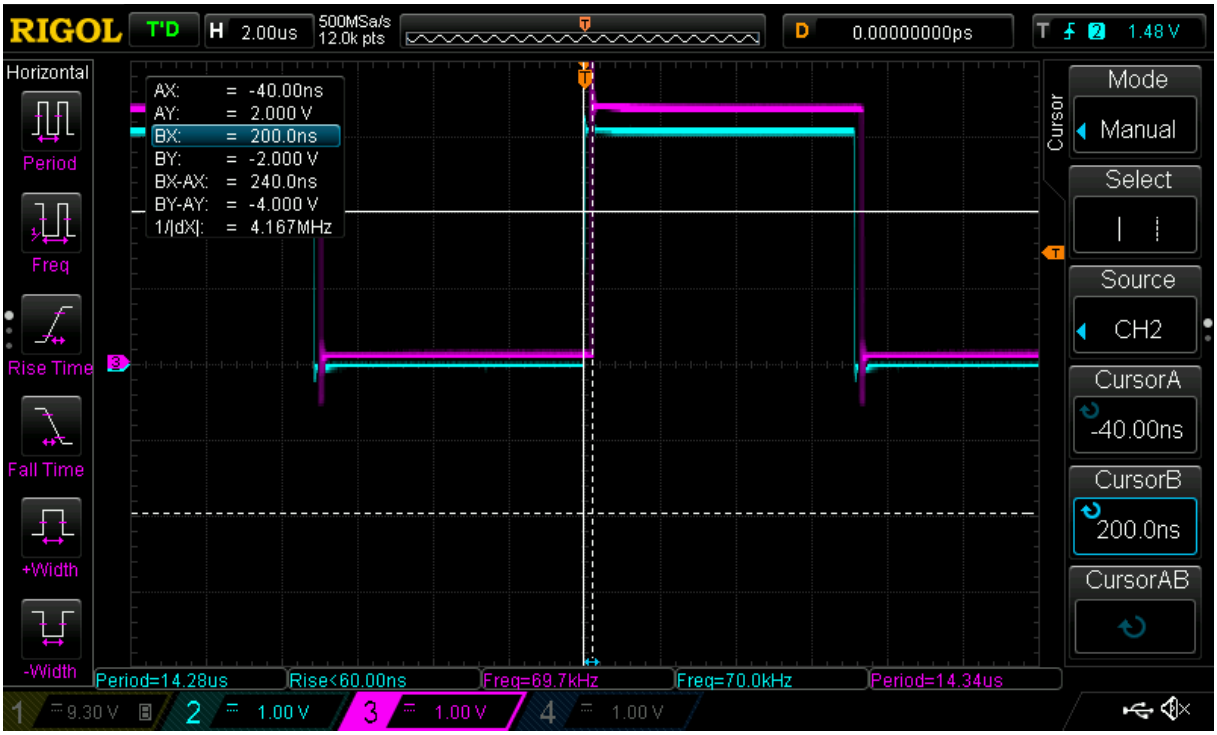
40 kHz	40.2 kHz	40 kHz	220 ns	190 ns
50 kHz	50.1 kHz	50 kHz	280 ns	190 ns
60 kHz	60.2 kHz	60 kHz	240 ns	200 ns
70 kHz	69.7 kHz	70 kHz	240 ns	200 ns
80 kHz	79.9 kHz	80 kHz	245 ns	200 ns
90 kHz	89.8 kHz	90 kHz	245 ns	200 ns
100 kHz	99.2 kHz	99.8 kHz	240 ns	200 ns
200 kHz	202 kHz	200 kHz	240 ns	200 ns
300 kHz	307 kHz	300 kHz	235 ns	200 ns
400 kHz	403 kHz	400 kHz	235 ns	210 ns
500 kHz	496 kHz	490 kHz	220 ns	210 ns
600 kHz	613 kHz	605 kHz	220 ns	210 ns
700 kHz	730 kHz	690 kHz	220 ns	220 ns
800 kHz	821 kHz	800 kHz	232 ns	224 ns
900 kHz	943 kHz	900 kHz	232 ns	228 ns

Na slici 4.10. prikazana je usporedba dobivenih rezultata kašnjenja signala za LinuxOS i LinuxRTOS. Iz grafa je vidljivo da se kod LinuxRTOS povećanjem frekvencije postepeno povećava kašnjenje nakon određene frekvencije. Kod LinuxOS kašnjenje je prilično ujednačeno i veće nego kod LinuxRTOS, a frekvencije počinju znatno odstupati nakon testne frekvencije od 100 kHz.



Sl. 4.10. *Grafički prikaz rezultata kašnjenja signala u odnosu na signal s funkcijskog generatora.*

Na slici 4.11. nalazi se prikaz pravokutnih signala gdje plavi signal predstavlja signal generiran funkcijskim generatorom, a ljubičasti signal predstavlja signal propušten kroz operacijski sustav koji se nalazi na Raspberry Pi 3.



Sl. 4.11. *Grafički izgled pravokutnih signala prikazanih na osciloskopu.*

Promatrajući dobivene rezultate vidljivo je da LinuxRTOS daje brži odziv, odnosno LinuxRTOS ima manja kašnjenja signala u odnosu na LinuxOS. Osim što ima manje kašnjenje signala vidljivo je da LinuxRTOS ima manja odstupanja frekvencije za razliku od LinuxOS što ga čini pogodnijim za rad s različitom periferijom koja zahtijeva frekvencijski ovisne upravljačke signale, kao primjerice upravljanje brzinom vrtnje motora i slično.

5. ZAKLJUČAK

Ugradbeni računalni sustav predstavlja sustav s specifičnom funkcijom u većem mehaničkom ili električnom sustavu. Zbog svoje znatno male veličine, niske potrošnje imaju skromne računalne mogućnosti, ali gotovo se svakodnevno koriste. Upotreba ugradbenih računalnih sustava je neograničena jer se na svakodnevno tržište uvode novi proizvodi koji koriste ugradbene računalne sustave na različite načine. Softver ugradbenih računalnih sustava može se definirati kao niz sustava koji se sastoji od operacijskog sustava koji uređaju omogućava izvršavanje zadanih poslova. Za izradu diplomskog rada koristi se Linux operacijski sustav koji je zbog svojih karakteristika vrlo rasprostranjen operacijski sustav, ali zbog svoje nedeterminističke prirode ne može se koristiti u operacijskim sustavima za rad u stvarnom vremenu. Izgradnja Linux operacijskog sustava za rad u stvarnom vremenu može se postići različitim modifikacijama Linux jezgre.

Cilj ovog diplomskog rada je izgradnja i usporedba dva tipa Linux operacijskih sustava Linux operacijski sustav za rad u stvarnom vremenu i standardni Linux operacijski sustav za Raspberry Pi 3 Model B. Nakon izgradnje Linux operacijskih sustava za lakšu usporedbu navedenih operacijskih sustava koriste se različita mjerenja, mjerenje mirovanja u sekundama, mjerenje mirovanja u mikrosekundama, mjerenje vremena potrebnog za zapisivanje teksta u odgovarajuću datoteku, mjerenje vremena potrebnog za čitanje teksta iz odgovarajuće datoteke, ciklično mjerenje i mjerenje brzine promjene signala na ulazima/izlazima opće namjene. Navedena mjerenja predstavljena su programima pisanim u C programskom jeziku.

Dobiveni rezultati svakog mjerenja prikazani su tablicama i na grafovima. Pregledom rezultata vidljivo je da Linux operacijski sustav za rad u stvarnom vremenu u određenom broju mjerenja osigurava točnije rezultate u odnosu na standardni Linux operacijski sustav.

LITERATURA

[1] T. Keser, predavanja s kolegija Ugradbeni računalni sustavi, studij: Računalno inženjerstvo, ak. god. 2018/2019, https://loomen.carnet.hr/mod/folder/view.php?id=221232 ,[06.08.2019]

[2] Embedded-systems, https://internetofthingsagenda.techtarget.com/definition/embedded-system [23.07.2019]

[3] Steve Heath, Embedded System Design. 2002. [23.07.2019]

[4] What is Linux, https://www.linux.com/what-is-linux [24.07.2019]

[5] What is a Real Time Embedded System, https://www.electronicshub.org/embedded-system-real-time-applications/ [25.07.2019]

[6] G. Martinović, predavanja s kolegija Računalni sustavi stvarnog vremena, studij: Računalno inženjerstvo, ak. God. 2018/2019, https://loomen.carnet.hr/pluginfile.php/314464/mod_resource/content/4/Predavanja/PR-1-ff.pdf [26.07.2019]

[7] Embedded System Software, https://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Embedded%20systems/Pdf/Lesson-28.pdf [26.07.2019]

[8] Real – Time Linux, https://www.linuxjournal.com/article/3980 [26.07.2019]

[9] Raspberry GPIO Pinout, https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html [27.07.2019]

[10] Raspberry Pi GPIO, https://www.raspberrypi.org/documentation/usage/gpio/ [27.07.2019]

[11] Raspberry Pi, https://www.raspberrypi.org/ [27.07.2019]

[12] Buildroot, https://buildroot.org/docs.html [27.07.2019]

[13] Bootlin Buildroot Training, https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf, [28.07.2019]

[14] Buildroot Images Repository, https://buildroot.org/download.html [28.07.2019]

[15] Linux Image Repository, https://www.kernel.org/ [28.07.2019]

[16] Real-Time Patch Repository, https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/ [28.07.2019]

[17] Buildroot, https://buildroot.org/downloads/manual/manual.pdf [29.07.2019]

[18] Real-Time Linux Evaluation, http://genderi.org/real-time-linux-evaluation-kalynnda-berens-grc.html [26.07.2019]

[19] L. Jeleković, predavanja s kolegija Sustavi za rad u stvarnom vremenu, http://www.zemris.fer.hr/~leonardo/srsv/skripta/_SRSV-skripta.pdf, [06.08.2019]

[20] Linux Operating System, https://www.techwalla.com/articles/types-of-linux-operating-systems, [06.08.2019]

[21] Types of Operating System, https://www.tutorialspoint.com/operating_system/os_types, [08.08.2019]

[22] Embedded Linux https://bootlin.com/doc/training/embedded-linux/embedded-linux-slides.pdf, [08.08.2019]

[23] R. Grbic, M. Herceg, predavanja s kolegija Linux u ugradbenim sustavima, studij: Računalno inženjerstvo, ak. God. 2018/2019, https://loomen.carnet.hr/course/view.php?id=7444, [08.08.2019]

[24] Real – Time Linux, https://slideplayer.it/slide/616975/, [09.08.2019]

[25] Raspoređivanje zadataka, http://www.zemris.fer.hr/~leonardo/os/dodatno/Osnovni_koncepti_OSa/OS_05_rasporedjivanje.pdf, [09.08.2019]

[26] Raspberry Pi GPIO Pinout, https://www.tomshardware.com/reviews/raspberry-pi-gpio-pinout,6122.html, [09.08.2019]

[27] QEMU emulator, https://qemu.weilnetz.de/doc/qemu-doc.html, [12.08.2019]

[28] Linux Scheduler, https://linux.die.net/man/2/sched_setscheduler [08.09.2019]

[29] WiringPi biblioteka, http://wiringpi.com/ [09.09.2019]

SAŽETAK

Tema ovog diplomskog rada je izgradnja i usporedba performansi dva tipa Linux ugradbenog operacijskog sustava za Raspberry Pi 3 Model B. U drugom poglavlju diplomskog rada detaljnije su opisani operacijski sustavi, koji su tipovi sustava za rad u stvarnom vremenu i koje su prednosti i nedostatci takvih sustava. U trećem poglavlju detaljnije je opisan Raspberry Pi 3 Model B i Buildroot alat. Buildroot alat koristi se radi lakše izgradnje Linux operacijskih sustava, te zbog potrebnih modifikacija kako bi se izgradio Linux operacijski sustav za rad u stvarnom vremenu. Četvrto poglavlje sadrži mjerenja koja se koriste radi lakše usporedbe Linux operacijskih sustava. Mjerenja se temelje na programima pisanim u C programskom jeziku i pokretani su na oba tipa Linux operacijskih sustava. Dobiveni rezultati mjerenja prikazani su u tablicama i na grafovima, gdje je vidljivo da Linux operacijski sustav za rad u stvarnom vremenu daje točnije rezultate u odnosu na standardni Linux operacijski sustav.

Ključne riječi: Raspberry Pi 3 Model B, Linux operacijski sustav, stvarno vrijeme, C programski jezik.

ABSTRACT

The topic of this graduate thesis is to build and compare the performance of two types of Linux embedded operating systems for the Raspberry Pi 3 Model B. The second chapter of the thesis describes operating systems in more detail, what are the types of real-time systems and what are the advantages and disadvantages of such systems . Chapter three describes the Raspberry Pi 3 Model B and Buildroot tools in more detail. The Buildroot tool is used to help build Linux operating systems more easily, and because of the necessary modifications to build a Linux operating system in real time. Chapter four contains measurements that are used to help compare Linux operating systems. The measurements are based on programs written in C programming language and run on both types of Linux operating systems. The obtained measurement results are shown in tables and graphs, where it is evident that the real-time Linux operating system produces more accurate results than the standard Linux operating system.

Keywords: Raspberry Pi 3 Model B, Linux operating system, real time, C programming language.

ŽIVOTOPIS

Tomislav Petrović, rođen 05.12.1993. u Vinkovcima, Hrvatska. Osnovnu školu „Matija Antun Reljković završava u Cerni, a srednju školu Tehničku školu „Ruđer Bošković“ završava u Vinkovcima. Nakon srednje škole, 2012. godine upisuje stručni studij na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek smjer Informatika. Nakon završenog stručnog studija i razlikovnih obveza 2017. godine upisuje diplomski studij smjer Računalno inženjerstvo.

Potpis:

_____

PRILOZI

Prilog 1 – programski kodovi izvršenih mjerenja pisanih u C programskom jeziku.

Prilozi se nalaze na DVD-u priloženom uz ovaj rad.