



Линукс курс 2018/2019

Увод у Линукс

Вежба 1

Распаковати архиву vezba1.tar.bz2. Позиционирати се у директоријум vezba1 и излистати његов садржај. Проверити да ли постоје сакривене датотеке. Променити назив директоријума tkst у tekst и у њега пребацити све датотеке са .txt екstenзијом (водити рачуна о сакривеним датотекама).

Сакривену датотеку из директоријума tekst ишчитати програмом more. Користити претрагу командом / у циљу проналажења речи Lasagna. Уколико more пријављује да није успео да нађе задату реч, то значи да се она не налази нигде у остатку датотеке који није приказан на экрану. Сада тражити реч one и видети шта се дешава. Отворити исту датотеку овог пута less програмом користећи синтаксу за замену програма у последњој команди. less програмом извршити исте претраге које су рађене у more програму и видети разлике. Када се стигне до краја датотеке користити претрагу уназад да се нађе реч spaghetti. Изаћи из less програма q командом.

Садржај директоријума tekst сместити у нову текстуалну датотеку са називом ls.txt. Учинити да сакривена датотека у tekst директоријуму више не буде сакривена и резултате потврдити поновним излисавањем ls командом. Сачувати нове резултате ls команде у ls2.txt датотеку. Командом diff упоредити ове две датотеке и тумачити добијене резултате. За боље разумевање diff команде консултовати приручник командом man diff.

Сместити тренутни директоријум (vezba1) на стек и прећи у директоријум tekst. Коришћењем условног извршавања ишчитати датотеку sardar.txt, а уколико извршавање ове команда није могуће направити датотеку са тим називом (команда touch, консултовати приручник за ову команду) и у њу уписати садржај датотеке sardar1.txt. Командом ls проверити шта се променило у директоријуму. Извршити поново претходну условну команду и видети да ли ће исход опет бити исти. Упоредити командом diff датотеке sardar.txt и sardar1.txt. Ишчитати првих 20 линија датотеке sardar2.txt. Уколико је пријављена грешка у вези права приступа, променити права приступа над датотеком sardar2.txt. Тестирати промену права извршавањем претходне



Линукс курс 2018/2019

команде за приказ првих 20 линија. Сада ишчитати последњих 10 линија датотеке sardar.txt тако да се укључи опција за приказивање нових промена у датотеци. Затим отворити нови терминал и из њега на крај датотеке sardar.txt уписати садржај датотеке sardar2.txt и пратити промене у раније отвореном терминалу. У другом терминалу ишчитати последњих 10 линија датотеке sardar2.txt и визуелно упоредити исписе у два терминала. Уколико су исписи исти, уписивање је добро извршено. У том случају затворити други терминал помоћу Ctrl+D. Прекинути tail програм помоћу Ctrl+C и потом додати садржај датотеке sardar3.txt на крај датотеке sardar.txt. Направити угњеждене директоријуме sardar/complete. У директоријум complete преместити sardar.txt под именом sardar_all.txt, а остале датотеке са sardar у имену преместити у sardar директоријум.

Ући у директоријум safe и ишчитати његов садржај ls командом. Ући у dir1 директоријум и поново ишчитати садржај помоћу ls. Шта се дешава? Улазити у директоријум dir1 све док се не дође до краја. После неколико cd команда, ишчитати садржај тренутног директоријума помоћу ls -l. Шта је специфично за dir1 директоријум? Вратити се у директоријум vezba1 скидањем овог директоријума са стека (команда rmdir). Командом pwd проверити тренутни директоријум. Прећи у директоријум safe и обрисати симболички линк. Обрисати и -o датотеку.

Вратити се у директоријум vezba1. Помоћу grep команде тражити појављивање речи santa у свим датотекама. Сада тражити реч Santa (вршити промене на претходној команди). Једном командом тражити и Santa и santa (не морају оба да се појављују у истој линији, дакле или једна или друга реч). Надовезати на претходну команду додатну претрагу, тако да се прикажу реченице које садрже речи santa и singh не правећи разлику између малих и великих слова. Сада искључити из приkaza реченице у којима се појављује и реч Banta. Консултовати приручник уколико постоје проблеми везани за grep команду и њене параметре.

Тражити реч money не правећи разлику између малих и великих слова у свим датотекама. Сада модификовати претходну команду додавајући додатне параметре тако да се прикажу и бројеви линија на којима је пронађена тражена реч. Приказати и по једну линију пре и после жељене линије додавајући параметре grep команди. Уклонити параметре за претходну и наредну линију, а



Линукс курс 2018/2019

додати параметар који ће обезбедити да се испишу само имена датотека у којима је нађена тражена реч, а не и саме линије. Резултат ове претраге проследити у нову команду (користити | xargs уколико није доволно) која ће извршити ls над овим датотекама. Уколико је исправност команде потврђена помоћу ls (исписане су само датотеке које је и grep пријавио) обрисати ове датотеке заменом ls са rm -r у претходној команди. Обрисати директоријум safe, командом за брисање празних директоријума.

Сортирати садржај declarations.h датотеке прво у обичном, а затим у обрнутом редоследу.

Направити копију текст директоријума помоћу rsync команде. Упоредити са командом diff ова два директоријума. Затим обрисати једну датотеку по избору из једног од директоријума и поново покренути diff команду, али овај пут њен излаз преусмерити у датотеку са називом diff.txt. Анализирати садржај ове датотеке.



Линукс курс 2018/2019

Увод у Линукс

Вежба 1

Направити директоријум vezba2 и у њему симболички линк vezba1 који ће показивати на vezba1 директоријум. Отворити текстуални едитор gedit из терминала. После отварања пребацити gedit у позадину одговарајућим командама за пребацивање програма у позадину како би терминал могао да се користи.

У gedit-у у новој датотеци искуцати следеће линије:

```
LOGD("Prvi ispis");
LOGE("LOGICAL CONDITION");
LOGW("WARNING, DON'T CHANGE LOGICAL IN PREVIOUS SENTANCE");
LOGI("INFO LOG");
```

Сачувати датотеку у директоријуму vezba1 под називом sed1.txt. Покренути нови процес gedit програмом, али овај пут додати одговарајуће параметре при покретању како би се процес покренуо као позадински. У другом gedit прозору искуцати следеће линије:

```
printf("Prvi ispis sa printf funkcijom");
printf("printing");
int printfile(msg) {
    LOGD("U printfile funkciji sam!");
    LOGD("%s", msg);
}
printfile("test...");
```

Ову датотеку сачувати у директоријуму vezba1 под називом sed2.txt.



Линукс курс 2018/2019

Излистати позадинске процесе одговарајућом командом, а затим помоћу команде `kill` затворити један од два `gedit` процеса (процес поред ког пише `done`). Други `gedit` процес затворити преко његовог `pid`-а. Информације о `pid`-у процеса пронаћи коришћењем `top` или `ps` команде. Покренути нови `gedit` процес и овај пут га затворити `xkill` командом.

Направити копију директоријума `vezba1` унутар `vezba2` директоријума коришћењем `rsync` команде. Позиционирати се у директоријум копије и `sed` командом у датотеци `sed1.txt` све позиве `LOG(X)` функције заменити позивима `ALOG(X)` функције (`LOGE` постаје `ALOGE`, `LOGD` постаје `ALOGD` итд). Обратити пажњу да се не промене секвенце `LOG` у осталим речима које не припадају позиву функције `LOG(X)`. Користити `nano` едитор за преглед датотеке после извршења резултата, у случају нежељених промена, `rsync` командом поново вратити копију у почетно стање и покушати поново. Исто урадити и са `sed2.txt` датотеком само што је овај пут потребно изменити `printf` у `printfile` позиве.

После ових измена упоредити садржаје директоријума `vezba1` и његове копије `diff` командом. Потом инсталирати пакет `meld`, и помоћу овог пакета покренути поређења ових директоријума, а такође и засебна поређења `sed1.txt` односно `sed2.txt` датотека из датих директоријума. Све инстанце `meld` програма покренути у позадини. После прегледа разлика у `meld`-у, затворити једном све `meld` прозоре командом `killall` и одговарајућим параметрима.

Запаковати копију `vezba1` директоријума `tar` командом, а затим направити нови директоријум `sed`, и у њега отпаковати само `sed1.txt` и `sed2.txt` датотеке из `tar` архиве. При паковању користити `bz2` компресију.

Вежба 2

Наставак вежбе се ради у паровима.

Потребно је за почетак направити два нова налога на сваком рачунару. Један налог је за корисника рачунара (у наставку вежбе се више не користи `rtrk`), а други за пару у вежби – на пример `rtrk-user1` и `rtrk-user2`. Налози треба да буду исти на оба рачунара. Налози се инсталирају преко менија `All Settings > User Accounts > Unlock`. Запамтити лозинке за управу додате кориснике. За ову вежбу потребно је инсталирати `ssh` сервер такође. Пакет који је потребно



Линукс курс 2018/2019

инсталирати је `openssh-server`. Име рачунара ће такође бити потребно и оно се може видети ишчитавањем датотеке `/etc/hostname`. Име рачунара је потребно приликом логовања преко `ssh` конекције на удаљени рачунар, а алтернатива је да се користи IP адреса рачунара. После успешно направљених налога и инсталираних потребних пакета, следећи корак је да се промени корисник , односно да се улогује са новонаправљеним налогом. Затим треба остварити `ssh` конекцију на рачунар свог пара у овој вежби. Даље треба да се изврше сви потребни кораци да се при отварању `ssh` конекције не мора уносити шифра. Шта све треба урадити?

Вежба 3

Покренути `vimtutor` и проћи тренинг за `vim` едитор.



Линукс курс 2018/2019

Упознавање са BASH скриптама

Циљ

Овладати BASH командним језиком и писањем BASH скрипти у смислу разумевања, измене постојећих или писања нових скрипти.

Исход

Након ове вежбе ћете моћи да:

- Разумете и мењате постојеће или пишете нове BASH скрипте.

ВЕЖБА 1

Отворите скрипту `bash_analiza1.sh` и прегледајте је. Разумите шта она ради. Заправо је то само један „параметризован“ пајп у командној линији.

Отворите скрипту `bash_analiza2.sh` и прегледајте је. Разумите шта она ради. Напишите нову скрипту по узору на ову у мало компактнијем и бољем маниру.

ВЕЖБА 2

Напишите скрипту која на позив приказује време и датум, излистава улоговане кориснике на рачунару, исписује податке о систему (процесор, меморија и време које је протекло од стартовања система). Након тога, сачувати ове информације у `log` датотеку.



Линукс курс 2018/2019

ВЕЖБА 3

Напишите скрипту која саму себе backup-ије, односно копира саму себе у датотеку са именом `backup.sh`.



Линукс курс 2018/2019

Упознавање са MAKE алатом и MAKEFILE датотекама

Циљ

У овој вежби ће бити приказан поступак за билдовање извршних датотека коришћењем `makefile` датотека.

Исход

Након ове вежбе ћете моћи да:

- Разумете и мењате постојеће или пишете нове `makefile` датотеке.

ВЕЖБА 1

Направите `makefile` датотеку у директоријуму `app` и додајте команде које ће омогућити превођење `cipher.c` датотеке и њено повезивање у извршну датотеку `app`. Објектне датотеке које настају током превођења је потребно сместити у директоријум `build`. Приликом повезивања искористити статичке библиотеке `ceasar`, `rot13` и `vigenere`.

ВЕЖБА 2

Проширите решење из претходног задатка тако да се статичке библиотеке билдују током креирања извршне датотеке `app`. Објектне датотеке које настају током превођења је потребно сместити у директоријум `build`. Приликом повезивања искористити статичке библиотеке из `build` директоријума.



Линукс курс 2018/2019

Raspberry Pi рачунар

Упознавање, повезивање и покретање Линукс оперативног система на различите начине

Увод

Рачунари се према фон Нојмановом моделу састоје од процесора, меморије и улазно-излазног подсистема који су међусобно повезани. У меморији се, поред података који се обрађују у процесору, складиште и програми састављени од низа елементарних инструкција. Током рада рачунара, подаци и програми се преносе између меморије и процесора. Данас, поред стоних, пресносних и осталих рачунара, постоје и различити вишенаменски уређаји који су прављени у складу са фон Нојмановом архитектуром, попут паметних телефона и таблета.

На тржишту се почетком 2012. године појавио рачунар Raspberry Pi који задовољава све критеријуме рачунара фон Нојмановог типа. Поседује чип BCM2835 са ARM11 процесором и RAM (енг. Random-Access Memory) меморију, као и могућност повезивања са осталим помоћним компонентама, чак и оним нестандартним преко GPIO (енг. General-Purpose Input/Output) порта. Оно што га чини интересантним је да је у питању рачунар опште намене приступачан по цени, малих димензија 8,6cm x 5,4cm x 1,7cm, са могућношћу прикључивања нестандартне опреме. Цена зависи од модела. Настанак рачунара Raspberry Pi имао је за циљ промоцију рачунарских наука код младих.

Настанак рачунара Raspberry Pi

Идеја о малом и приступачном рачунару јавила се 2006. године, када су Роб Мулинс, Ебен Аптон, Џек Ланг и Алан Мајкрофт са Универзитета у Кембриџу постали забринuti нивоом предзнања студената који су се пријављивали за рачунарске науке. За разлику од 1990. године када је већина кандидата имала солидно предзнање из области програмирања, 2000. године је просечан кандидат имао мало знања о програмирању.

Закључили су да се формирала армија информатичара са врло мало практичног програмерског знања потребног за исправно решавање конкретних изазова. Они су имали искуства у коришћењу кућног рачунара или конзоле за игру и забаву. Искуство стечено свакодневним корисничким радом на рачунару давало им је лажну слику о личном знању из области рачунарских наука. Претходне генерације



Линукс курс 2018/2019

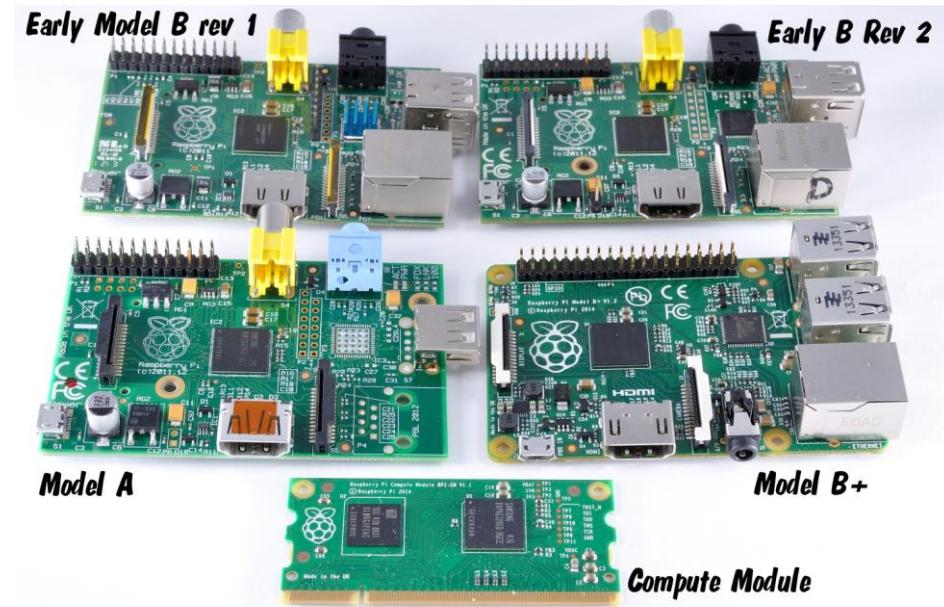
училе су програмирање на легендарним Спектрум, Комодор64 или Амига рачунарима, који су осим могућности забаве обезбеђивали и солидно окружење за писање првих програма. Због тога су дошли на идеју да направе нешто што је ученицима приступачно по цени, а пружа окружење спремно за писање програма. Од 2006. до 2008. године дизајнирано је неколико верзија од којих је настао рачунар Raspberry Pi. Када су се 2008. године појавили процесори који су могли да се користе за мобилне уређаје, приступачни по цени и довољно снажни да обезбеде добру подршку графичком окружењу, постало је извесно да ће пројекат заживети.

Када се на тржишту појавио први примерак рачунара Raspberry Pi, својим изгледом и могућностима привукао је велику пажњу, како у области обуке програмера, тако и код ентузијаста. Пошто је сам уређај изазвао велику пажњу, на тржишту постоји много адекватне пратеће опреме, а на Интернету је објављено мноштво конкретних објашњења и практичних упутстава за коришћење.

Модели рачунара Raspberry Pi

Историјски гледано, постоји 5 модела рачунара Raspberry Pi. Нема суштинске разлике у архитектури између ових модела. Нови модели су настали као одговор на добро формулисане захтеве за променама на постојећим моделима, добијене од стране самих корисника. Разлике се огледају у типу процесора, количини постојеће RAM меморије, у броју и врсти прикључака који се налазе на плочи уређаја и распореду компоненти на плочи. Сама чињеница да постоје потребе за променама на моделима указује да је настанак једног таквог рачунара био прави потез. Корисницима је остављена могућност да сами одаберу модел који им по особинама и функционалности највише одговара у складу са конкретним потребама.

Линукс курс 2018/2019



Слика 1 Модели рачунара Raspberry Pi

Модел Б ревизија 1 се појавио у фебруару 2012. године. На плочи има чип BCM2835 на коме се налазе: ARM11 процесор на 700 MHz и VideoCore IV GPU (енг. Graphics Processing Unit). Поред чипа на плочи се налазе 256MB RAM меморије, HDMI (енг. High-Definition Multimedia Interface) излаз, композитни RCA (енг. Radio Corporation of America) видео, стерео аудио 3,5mm, два USB (енг. Universal Serial Bus) и један мрежни прикључак.

У септембру 2012. године појавио се **модел Б ревизија 2**, који у односу на модел Б ревизија 1 има два пута више RAM меморије. Између ова два модела постоји мала разлика у распореду пинова на GPIO порту о чему се мора водити рачуна приликом писања програма за одређени модел. Новији модел је нашао употребу у образовању и као подршка за озбиљније пројекте у кући и лабораторији.

Линукс курс 2018/2019



Слика 2 Модел B rev2

Модел А је настао у фебруару 2013. године. Он је поједностављена, а самим тим и јефтинија верзија претходних модела. Поседује само 256MB RAM меморије, један USB прикључак и нема мрежни прикључак. Због малог броја прикључака доволјно му је и слабије напајање, те се најчешће користи за пројекте у којима се обрађује конкретан проблем. На пример, уколико је потребан кућни медија центар повезан са телевизором, модел А са USB WiFi бежичним прикључком је сасвим довољан за реализацију таквог пројекта. Недостатак другог USB прикључка могуће је превазићи USB раздељником (енг. hub) који има сопствено напајање, а недостатак мрежног прикључка употребом USB WiFi бежичног прикључка. Једини изазов који преостаје је количина расположиве RAM меморије која је константна и не може се повећати.



Слика 3 Модел А

Линукс курс 2018/2019

Модел Б+ се појавио у јулу 2014. У односу на претходни модел Б ревизија 2 уместо 26 пинова на GPIO порту има 40 пинова, уместо два има четири USB 2.0 прикључка, уместо SD користи микро SD меморијску картицу. Услед замене одређених компоненти троши мање струје тако да је продужено време рада са батеријом и има побољшану толеранцију на пад напона. Композитни видео и стерео аудио прикључак су спојени у један, што је елегантније и практичније решење, а уједно је и побољшан квалитет аналогног звучног сигнала. Распоред прикључака је такав да се каблови који воде до њих сада налазе само са две стране уређаја. На плочи има четири места (рупе за шрафове) намењена монтажи.

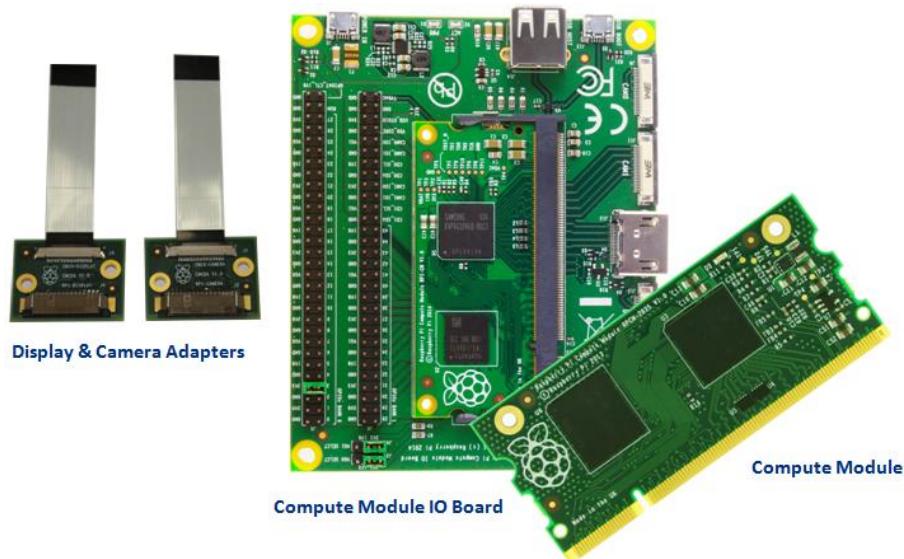


Слика 4 Модел Б+

Compute module развојни пакет се појавио у јуну 2014. године и намењен је свима који желе да изађу из оквира који нуди стандардни рачунар Raspberry Pi. Састоји се из картице и основне плоче која повезује картицу са периферним уређајима. На картици се налазе процесорски чип BCM2835, 512MB RAM и 4GB eMMC (embedded Multi-Media Controller) меморије. eMMC меморија је еквивалентна меморијској картици истог капацитета која се користи на стандардним Raspberry Pi рачунарима. Картица задовољава стандард DDR2 SODIMM (Double Data Rate 2, Small Outline Dual In-line Memory Module) прикључка преко којег се повезује са плочом на којој се налазе прикључци за остале уређаје. У развојном пакету, поред цомпјуте модул картице добија се и једна основна штампана плоча димензија 67,6 mm x 30 mm која, поред прикључака за периферне уређаје, на себи има стандардни DDR2 SODIMM прикључак преко кога се омогућава рад са картицом. За цомпјуте модул картицу је могуће конструисати

Линукс курс 2018/2019

посебну плочу на којој се налазе приклучци за периферне уређаје, једини услов је да на плочи постоји место за DDR2 SODIMM картицу. Наведени модел је најподеснији за осмишљавање и конструкцију малих уређаја специјалне намене.



Слика 5 Compute module

Raspberry Pi 2 Модел Б представља **другу генерацију платформе** која се појавила у фебруару 2015. Нови модел је донео и значајно побољшање перформанси. Raspberry Pi 2 је чак 6 пута бржи од свог претходника. Нови Raspberry Pi 2 Модел Б је истог формата као претходни Raspberry Pi модел Б+ али са дупло више RAM меморије и знатно бржим процесором. Овај рачунар величине кредитне картице је у могућности да обавља многе послове као и стони рачунар, на пример да покрене програме за табеларне калкулације, обраду текста, преко њега може да се сурфује интернетом и чак одигра нека игра у HD формату. Подржава неколико верзија Линукса као и бесплатну верзију Виндоус 10 оперативног система (не пуну верзију као за стони рачунар).

Линукс курс 2018/2019



Слика 6 Raspberry Pi 2 Модел Б

Raspberry Pi Zero се појавио у новембру 2015 као два пута мањи модел од старог модела A+ и два пута бољим карактеристикама, изузетно пиступачан поцени од \$5. Иако са ограниченим могућностима, ипак је доволјан, а свакако приуштив за било који пројекат:

1Ghz, Single-core CPU

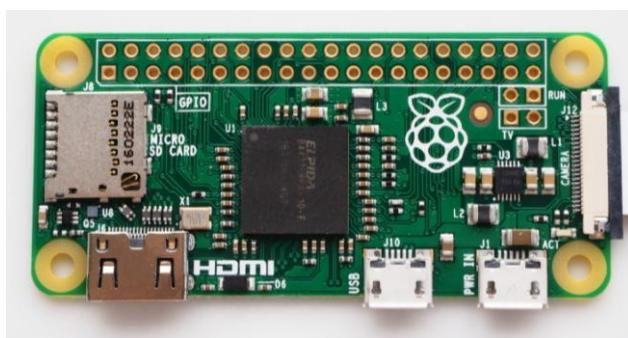
512MB RAM

Mini HDMI and USB On-The-Go ports

Micro USB power

HAT-compatible 40-pin header

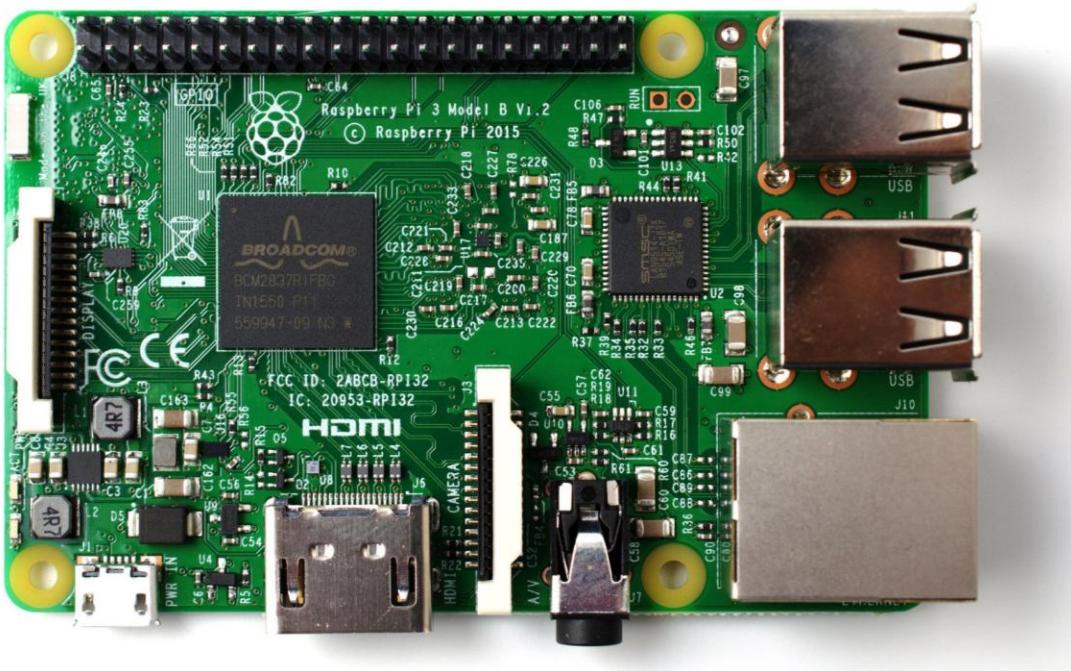
Composite video and reset headers



Слика 7 Raspberry Pi Zero

Линукс курс 2018/2019

Raspberry Pi 3 Модел Б представља трећу генерацију платформе која се појавила у фебруару 2016. Нови модел је донео мање значајна побољшања: 1.2GHz 64-bit quad-core ARMv8 процесор, интегрисану подршку за 802.11n Wireless LAN и Bluetooth 4.1.



Слика 8 Raspberry Pi 3 Модел Б

Тајни састојак и даље популарног модела Raspberry Pi 2 плоче је Broadcom BCM2836 процесор, ARMv7 Quad Core Processor System-on-Chip, који ради на 900MHz а поседује и Videocore 4 GPU. GPU (графички процесор) подржава OpenGL ES 2.0, хардверски убрзани OpenVG, декодирање 1080p30 H.264 висококвалитетног видеа и способан је да обради 1Gpixel/s, 1.5Gtexel/s или да извршава рачунске операције брзином од 24 GFLOPs-а за различите намене. То нпр. омогућава повезивање RPi2 на HDTV и гледање видео BlueRay квалитета користећи H.265 са протоколом од 40MBits/s.

Највећа промена на RPi2 у односу на претходнице је поред новог процесора и повећање RAM меморије са 512MB на 1GB. И даље се користи микро SD картица као системски диск. Подржане су и картице веће од 4GB, мада ће већина Линукс дистрибуција радити савршено и на 4GB.

Raspberry Pi 2 има четири USB порта (миш, тастатура, итд.). Уколико је за рад потребно још додатних USB портова, може се употребити USB разделник у сврху проширења могућности. Препоручује се употреба USB разделника са додатним



Линукс курс 2018/2019

напајањем како се не би преоптеретио извор напајања на RPI2 плочи. Напајање плоче се обезбеђује преко микро-USB порта на који се може прикључити спољашњи USB адаптер или пуњач за паметни телефон. Препоручује се адаптер који може да обезбеди минимум 1A а идеално 2A или више ако се жели да повеже више додатних уређаја на USB.

Поврх свега, периферије ниског нивоа на Pi плочи омогућавају разна експериментисања. На располагању је и 40-пински конектор са растером од 2.54mm преко кога је омогућен приступ ка 27 GPIO пинова, UART, I2C и SPI порту, као и напајањима од 3.3V и 5V. Овај конектор је потпуно исти као на Б+ плочама, стога се без икаквих измена могу користити и shield-ови пројектовани за Б+.

Техничке карактеристике:

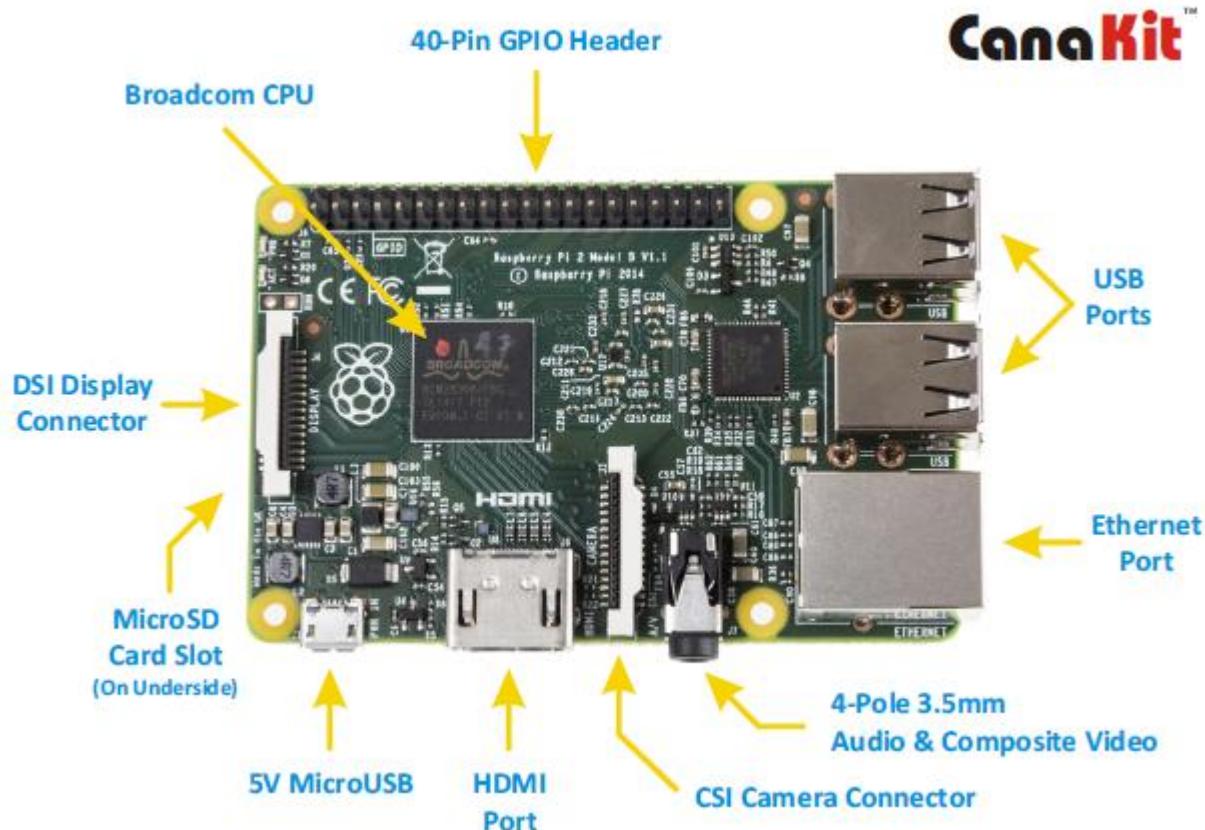
- Broadcom BCM2836 SoC
- 900MHz ARMv7 Quad Core Processor SoC
- Broadcom VideoCore IV GPU
- 1 GB RAM
- 4 x USB2.0 Ports with up to 1.2A output
- Expanded 40-pin GPIO Header
- Video/Audio Out via 4-pole 3.5mm connector, HDMI, or Raw LCD (DSI)
- Storage: microSD
- 10/100 Ethernet (RJ45)
- CSI camera connector
- Low-Level Peripherals:
 - 27 x GPIO
 - UART
 - I2C bus
 - SPI bus with two chip selects
 - +3.3V
 - +5V
 - Ground
- Напајање: 5V @ 600 mA преко MicroUSB кабла или GPIO конектора
- Подржава Raspbian, Windows 10, Debian GNU/Linux, Fedora, Arch Linux, RISC OS, итд.

Имајући у виду све моделе који су доступни на тржишту, може се закључити да сваки модел понаособ има своје предности и мане. Модел А је мали потрошач и самим тим интересантан за уско-специјализоване пројекте. Сви модели Б типа су универзални модели за обуку и експериментисање са могућностима које пружа рачунар Raspberry Pi. Compute module развојни пакет је намењен стручњацима из

Линукс курс 2018/2019

области електронике. Raspberry Pi Zero је по цени далеко најприступачнији, а и даље завидних перформанси на плочици изузетно малих димензија.

У свим практичним примерима у раду користиће се рачунар Raspberry Pi 2 модел Б.



Слика 9 Raspberry Pi 2 модел Б

Додатна опрема

Поред жељеног модела рачунара Raspberry Pi на коме се налазе чип са процесором и RAM меморија, за рад потребна је и додатна опрема која задовољава одређене стандарде.

1. **Кућиште** за рачунар Raspberry Pi може да се набави одвојено или у комплету заједно са плочом и на тржишту постоји широка понуда тако да се може одабрати оно које највише одговара потребама. Основна намена му је да заштити рачунар од спољашњих утицаја, обезбеди добру вентилацију и приступ свим потребним прикључцима. Осим комерцијалних, која могу бити једноставна, елегантна, шарена или практична у употреби су и уникатна



Линукс курс 2018/2019

кућишта направљена од приручних материјала - LEGO коцкица, пластичне или дрвене кутије...

2. **Електрично напајање** мора да задовољи следеће услове: да даје напон од 5V једносмерне струје, да даје струју већу од 700mA и да има микро USB прикључак. Уколико је потребно да користимо више USB прикључака истовремено, пожељно је јаче напајање рецимо 1,5A или чак 2A. Ове карактеристике задовољавају напајања која се користе за мобилне телефоне. Струја се може преузимати из електричне мреже, са USB порта другог рачунара или из батерије која се на тржишту може набавити као додатно напајање за мобилни телефон.
3. **Меморијска картица** треба да буде капацитета најмање 4GB. Препорука је да се користи картица капацитета 8GB и најмање класе 4. За моделе A, Б ревизија 1 и Б ревизија 2 се користи SD, а за новији модел B+ као и другу генерацију платформе RPi2 микро SD меморијска картица.
4. **Raspberry Pi камера** је специјално дизајнирана камера намењена за Raspberry Pi рачунаре која се прикључује на CSI (Camera Serial Interface) прикључак на плочи уређаја. Постоје две варијанте камера: прва намењена дневном снимању и друга, Pi Noir камера, намењена дневном и ноћном снимању. Pi Noir камери је уклоњен инфрацрвени филтер што јој у ноћним условима повећава осетљивост и омогућава снимање са инфрацрвеним осветљењем. Других разлика између ова два модела нема. Према официјелној Интернет страници по цени од 25\$ добија се камера тежине 3g и димензија 25mm x 20mm x 9mm. Слика добијена овом камером има резолуцију 5 Mpixel, а видео 1080p30, 720p60 и 640x480p60/90. Сензор на камери има резолуцију 2592 x 1944 пиксела, а област којом слика је димензија 3.76mm x 2.74mm. Формати слика који се добијају овом камером су JPEG, JPEG+RAW, GIF, BMP, PNG, YUV420, RGB888, а формати видео записа су raw и h.264. Детаљне карактеристике ове камере наведене су на званичном сајту. Коришћењем ове камере добија се на брзини, јер је у *Raspbian* оперативном систему предвиђено да се омогућавањем рада ове камере део RAM меморије резервише за рад GPU. Поред ове камере, могуће је користити IP или стандардну USB камеру.

Линукс курс 2018/2019

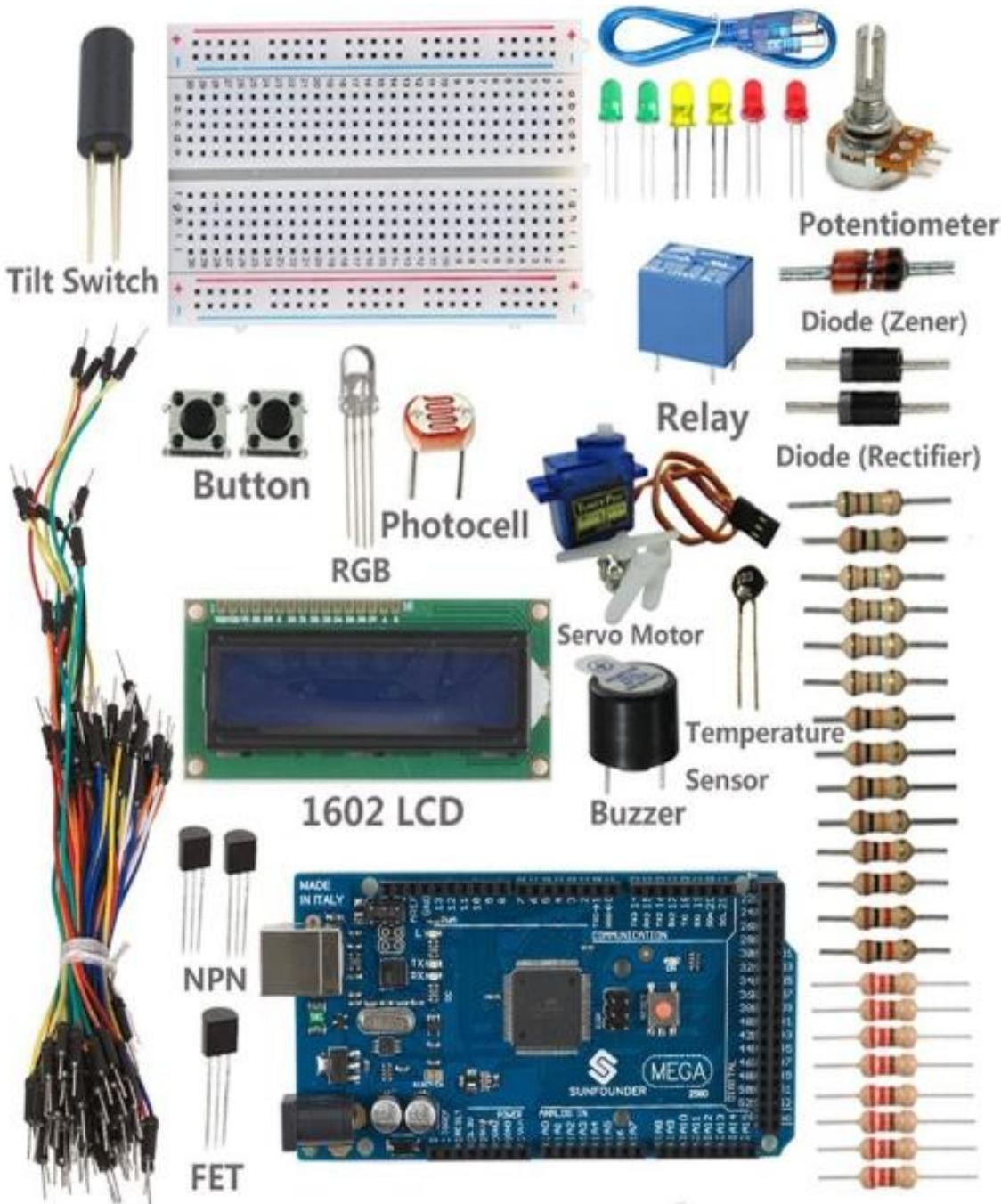


Слика 10 Raspberry Pi 2 Модел Б комплет

Остала рачунарска опрема нема посебних захтева, осим да постоји могућност физичког повезивања са рачунаром Raspberry Pi преко одговарајућег прикључка. Повезивањем на рачунарску мрежу, мрежним каблом или бежичним WiFi USB прикључком, рачунар Raspberry Pi добија своју IP адресу и омогућен је приступ до њега са осталих уређаја у мрежи, тачније понаша се као и сваки други рачунар. Уколико се рачунар Raspberry Pi користи као самосталан рачунар, односно, потребан је директан рад на њему у жељеном окружењу, од додатне опреме су потребни екран, USB тастатура и миш. Повезивање са екраном се обавља преко HDMI кабла или одговарајућег адаптера. USB тастатура и миш могу бити физички одвојени, а могуће је користити и комерцијалне комбинације са идејом да се искористи само један USB порт.

Идеја на основу које је настао рачунар Raspberry Pi је била да се првенствено млади заинтересују за конкретне проблеме из области рачунарских наука. Осим стандардне рачунарске опреме, на рачунар Raspberry Pi је могуће повезати преко GPIO порта и неку другу опрему из области електронике. За ту намену на тржишту постоји и почетнички комплет по приступачној цени, за рад са рачунаром Raspberry Pi, намењен прављењу прототипова решења конкретних проблема. Када се направи прототип решења, могуће је програмски контролисати понашање датих компоненти.

Линукс курс 2018/2019



Слика 11 Комплет додатне опреме

Упутство за коришћење

Сви студенти имају обавезу да опрему користе у складу са смерницама за употребу датим од производјача опреме. Упутство за коришћење као и



Линукс курс 2018/2019

повезивање периферних уређаја може да се пронађе у Raspberry Pi 2 развојном комплету под називом “QUICK START GUIDE”.

Припрема окружења

Преузимање датотека и директоријума који ће надаље бити коришћени

Инсталација података за вежбе

За предстојеће вежбе на курсу, припремљен је сет података (image-и језгра, конфигурације језгра, root filesystems, и др), а можете га преузети на један од два начина:

- Преузимањем GIT репозиторијума са GitHub-a

```
sudo apt-get install git  
cd  
git clone --depth=1 https://github.com/rtrk/linux-kernel-labs.git
```

- Преузимањем tarball архиве са путање

\\samba03\\nastavni_materijali\\aadsp1\\linux-kernel-labs.tar.bz2

Преузмите tarball и поставите га у home директоријум, а потом распакујте у терминалу:

```
cd  
tar xvjf linux-kernel-labs.tar.bz2
```

Распакујете Root Filesystem са:

```
sudo tar xvjf linux-kernel-labs/modules/rootfs.tar.bz2 -C linux-kernel-labs/modules/nfsroot
```

Поставите на следећи начин власнике појединачних датотека:

```
sudo chown -R rtrk.rtrk linux-kernel-labs/modules/nfsroot/root
```

Подаци за вежбе се сада налазе у вашем home директоријуму, унутар директоријума linux-kernel-labs. Ту се налазе директоријуми који садрже различите податке неопходне за сваку од предстојећих вежби. Овај директоријум



Линукс курс 2018/2019

ће такође бити коришћен и као радни простор за сваку вежбу, тако што ће датотеке које направите за време сваке вежбе бити чуване посебно.

Сада сте спремни за почетак практичних вежби.

Инсталација додатних пакета

Слободно инсталирајте друге пакете који су вам потребни у развојном окружењу. Генерално, препоручујемо инсталацију вама омиљеног текст едитора и конфигурацију истог према вашем укусу. Омиљени текст едитори за људе који се баве embedded Линуксом су свакако Vim и Emacs, али постоји и мноштво других могућности, попут GEdit, Qt Creator, CodeBlocks, Geany, итд.

Вреди поменути да Ubuntu иницијално долази са веома ограниченој верзијом едитора vi. Стога, уколико желите да користите vi едитор, предложамо да користите верзију богатију опцијама инсталирањем пакета vim.

Додатна упутства

Може бити корисно у свим вежбама:

- Прочитајте пажљиво упутства и савете. Многи погреше или губе време јер су пропустили објашњење или савет.
- Увек читајте поруке о грешкама, посебно прву. Многи се заглаве на веома једноставним грешкама само зато што су навели лошу путању до датотеке и нису обратили пажњу на одговарајућу поруку о томе.
- Немојте стајати блокирани на чудном проблему више од 5 минута. Покажите проблем колегама или асистенту.
- root корисника би требало да користите само за операције које захтевају супер привилегије, као што су mount-овање фајл система, учитавање модула језgra, измена власника датотеке, конфигурисање мреже. Већину регуларних задатака (преузимање, распакивање, компајлирање, итд) је могуће завршити као регуларан корисник.
- Ако грешком покрећете команде из root shell-а, ваш регуларни корисник можда више неће бити у могућности да користи одговарајуће датотеке. У том случају користите chown -R команду да нове датотеке вратите регуларном кориснику.

Линукс курс 2018/2019

Нпр: chown -R myuser.mygroup <path>

- Ако користите Gnome терминал (основни емулатор терминала на Ubuntu 16.04), можете користити вишеструке табове да бисте имали више терминала у истом позору. Ако не постоји опција за прављење новог табе, можете то урадити комбинацијом тастера [Ctrl] [Shift] [t].

Припрема Raspberry Pi рачунара

Циљ: успостављање комуникације са плочом и конфигурација bootloader-а

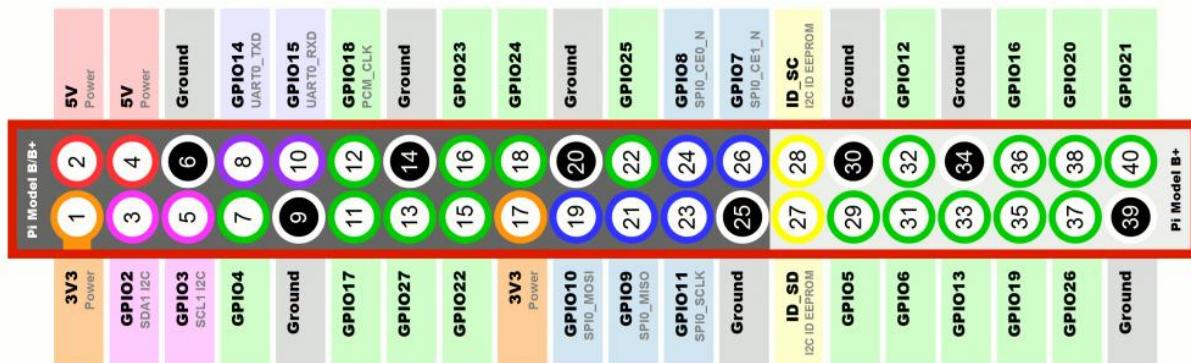
Након ове вежбе моћи ћете да:

- Приступите плочи преко серијске линије
- Конфигуришете U-boot bootloader и tftp server на вашем рачунару да бисте пребацивали датотеке кроз tftp.

Упознавање са плочом

Доступне периферије и конекторе на Raspberry Pi 2 рачунару приказује Слика 9. У истом поглављу (Модели рачунара Raspberry Pi) се налазе и детаљне карактеристике и спецификација овог модела.

Распоред GPIO пинова на 40-пинском конектору приказује Слика 12.



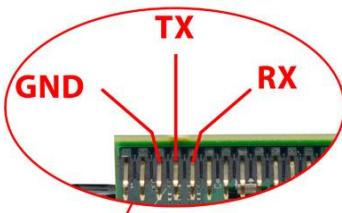
www.raspberrypi-spy.co.uk

Слика 12 Распоред GPIO пинова на 40-пинском конектору Raspberry Pi рачунара

Линукс курс 2018/2019

Успостава серијске комуникације са плочом

Серијски конектор Raspberry Pi рачунара се налази на 40-пинском конектору, као што је приказано на слици испод (Слика 13).



Слика 13 Распоред пинова серијског порта на конектору Raspberry Pi рачунара

ДОК ЈЕ ПЛОЧА ИСКЉУЧЕНА, користећи специјалан „USB to Serial“ адаптер (Слика 14) који сте добили од асистента, прикључите жице на приказани конектор на следећи начин: GND (црна) прикључите на pin означен са GND (pin 6, Слика 12), а RX (бела) и TX (зелена) жице на пинове TX и RX на плочи (пинови 8 и 10, Слика 12). Увек водите рачуна да TX жицу кабла (адаптера) спојите на RX pin плоче и обрнуто, који год кабел или плочу користите.



Слика 14 „USB to Serial“ адаптер, распоред пинова (жица)

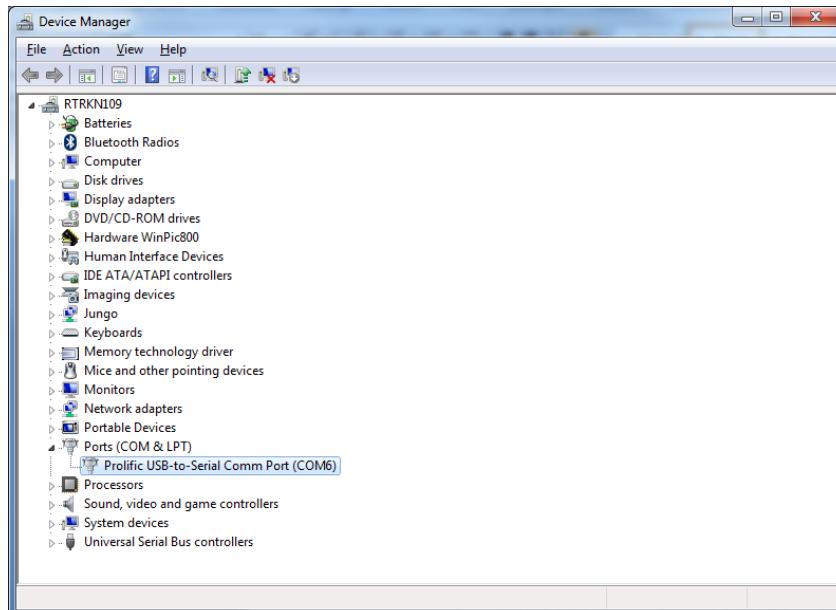
Изглед повезаног „USB to Serial“ адаптера и Raspberry Pi плоче приказује Слика 15.

Линукс курс 2018/2019



Слика 15 Повезивање „USB to Serial“ адаптера и Raspberry Pi плоче

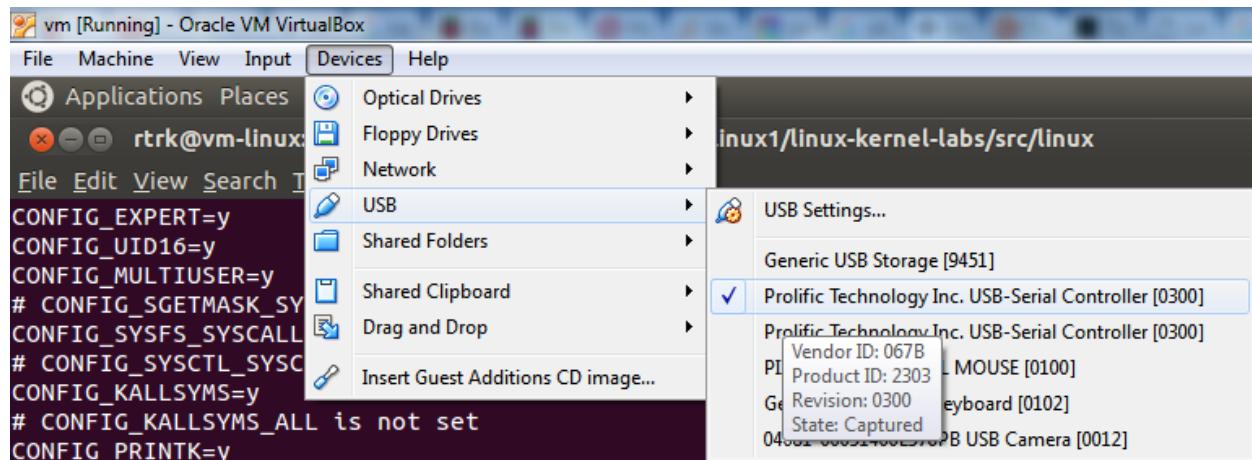
Када је „USB to Serial“ адаптер повезан на рачунар, нови серијски порт би требало да се појави као /dev/ttyUSB0, уколико користите Линукс. Уколико на рачунару имате Windows, адаптер би требало да се види у Device Manager-у, Слика 16.



Слика 16 Приказ „USB to Serial“ адаптера у Device Manager-у (Windows)

Линукс курс 2018/2019

У случају да радите на виртуелној машини, а домаћин има Windows оперативни систем, потребно је још да омогућите госту (ВМ) да користи прикључени адаптер, Слика 17. Након тога, у ВМ добијате исту ситуацију као када користите на рачунару Линукс (нови серијски порт би требало да се појави као /dev/ttyUSB0), односно Windows.



Слика 17 Прослеђивање „USB to Serial“ адаптера госту на виртуелној машини

Напомена: Да би ВМ систем автоматски препознао ваш USB уређај потребно је додати филтер уређаја. У опцијама Devices-> USB -> USB Settings... додати нови USB филтер са именом уређаја који омогућује USB на серијску везу. Након овога, следећи пут када се покрене ВМ потребно је само закачити назначени USB уређај и ВМ систем ће га аутоматски препознати и додати.

Да бисте комуницирали са плочом кроз серијски порт, инсталирајте најпре програм за серијску комуникацију попут picocom:

```
sudo apt-get install picocom
```

Ако покренете команду ls -l /dev/ttyUSB0, можете видети да само root и корисници који припадају dialout групи имају права приступа датотеци ради читања и писања. Стога, треба додати вашег корисника у dialout групу:

```
sudo adduser $USER dialout
```

Сада је још потребно да урадите log out и log in поново да бисте омогућили новој групи да буде свуда видљива.



Линукс курс 2018/2019

Сада можете покренути picocom -b 115200 /dev/ttyUSB0, да бисте започели серијску комуникацију на /dev/ttyUSB0, са baudrate параметром постављеним на 115200. Уколико желите да напустите picocom, притисните [Ctrl] [a] праћено са [Ctrl] [x].

За сада не би требало да буде ништа на серијској линији, с обзиром да плоча није укључена. Сада је време да се укључи плоча прикључивањем mini-USB кабла који сте добили од асистента (са другим крајем прикљученим на ваш рачунар или USB напајање).

Проверите које поруке добијате на серијској линији. Ако је све коректно урађено, требали бисте да видите U-boot испис на серијској линији.

Интеракција са Bootloader-ом

Ресетујте плочу. У picocom терминалу треба да видите како се редом подиже U-boot, па Линукс кернел и на крају сам Линукс. По завршетку треба да добијете промпт за пријаву на систем са именом buildroot. Користите налог root, без лозинке.

```
Welcome to Buildroot
```

```
buildroot login: root
```

```
buildroot:~#
```

Ресетујте поново плочу и притисните било који тастер да зауставите одбројавање U-boot-а када оно почне. Тада би требало да видите U-boot промпт:

```
U-Boot>
```

Сада можете да користите U-boot. Покрените help команду да видите листу доступних команда и уверите се да имате најмање верзију 2013.10.

Покретање Линукс оперативног система

Следећи корак је конфигурација U-boot-а и вашег рачунара тако да се омогући пребацање датотека, као што су kernel image и Device Tree Binary (DTB), коришћењем TFTP протокола кроз Ethernet кабел.

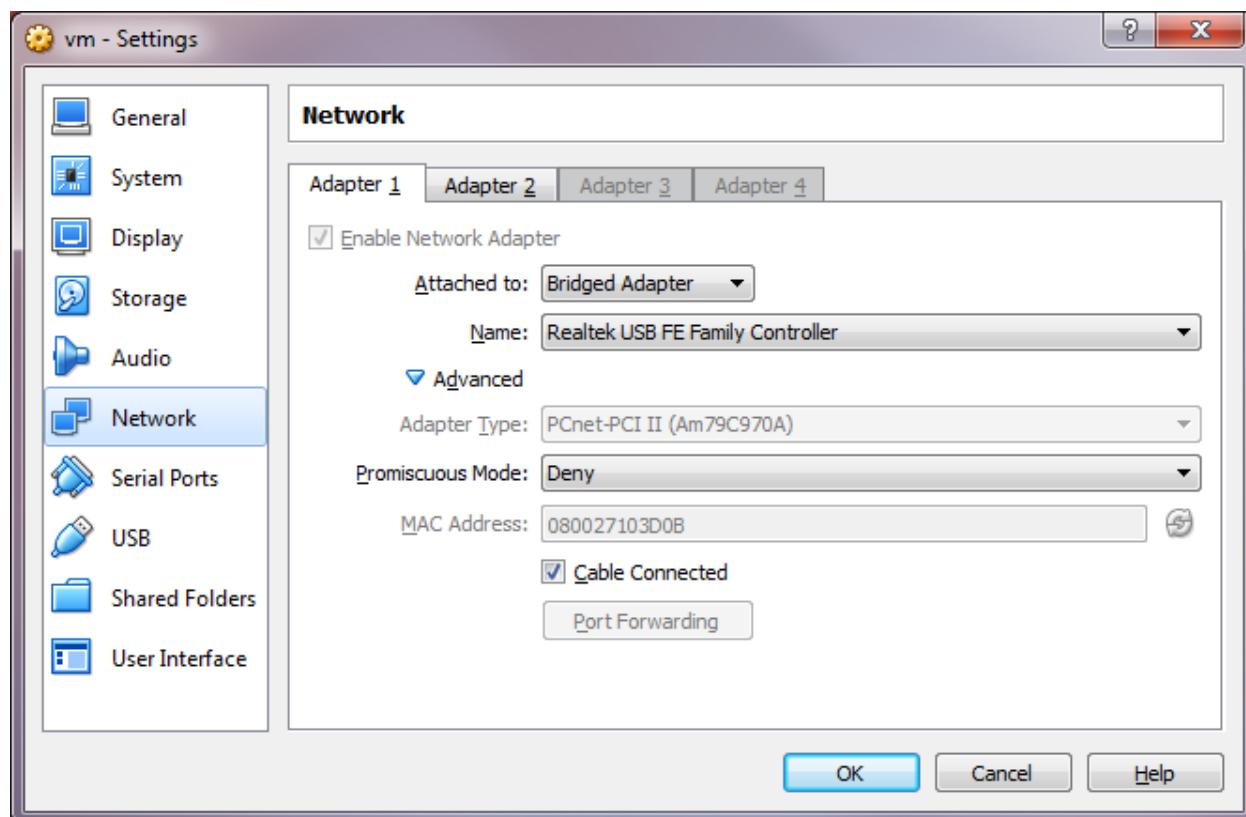
Линукс курс 2018/2019

За почетак, инсталирајте TFTP server на ваш рачунар:

```
sudo apt-get install tftpd-hpa
```

(Опција 1) Повезивање Raspberry Pi директно са рачунаром

Мрежним каблом, повежите Ethernet порт Raspberry Pi плочице са портом на рачунару. Уколико рачунар већ користи порт за конекцију на мрежу, од асистента ћете добити USB Ethernet адаптер. Уколико радите у виртуелној машини, потребно је и прикључити нови уређај (адаптер) на виртуелну машину, Слика 18.



Слика 18 Прикључивање мрежног адаптера на виртуелну машину

Нова мрежна спрега, вероватно са ознаком eth0 или eth1 би требало да се појавила у Линукс систему.

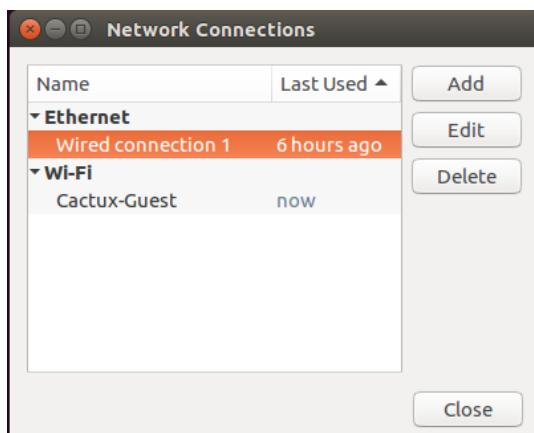
Да бисте конфигурисали мрежну спрегу на рачунару, покрените Network Manager на рачунару и одaberите Edit Connections опцију, Слика 19.

Линукс курс 2018/2019



Слика 19 Покретање алата за подешавање мрежне спреге

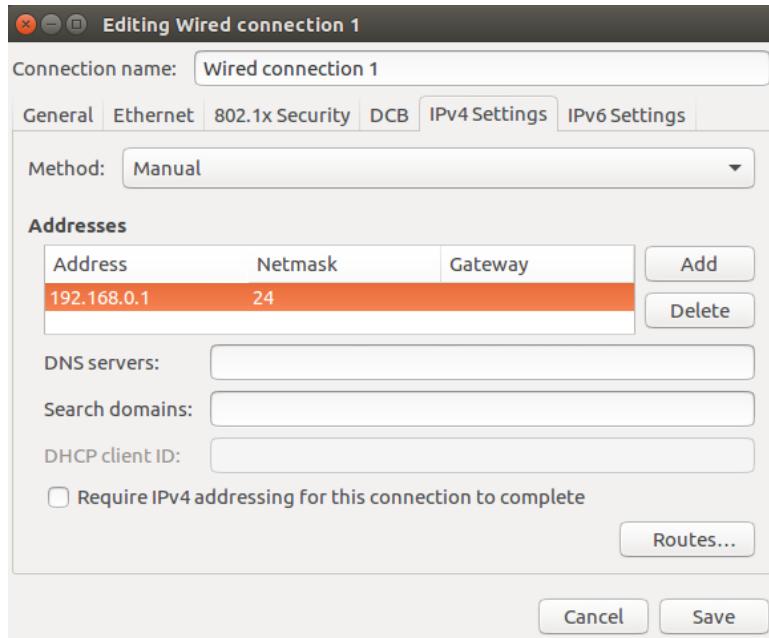
Одаберите нову конекцију преко кабла:



Слика 20 Нова мрежна конекција преко кабла

У картици IPv4 Settings, одаберите Manual као метод да бисте омогућили да спрена користи статичку IP адресу, попут 192.168.0.1 (наравно, водите рачуна да ова адреса припада другом мрежном сегменту у односу на главну компанијску мрежу на коју је рачунар већ повезан), Слика 21.

Линукс курс 2018/2019



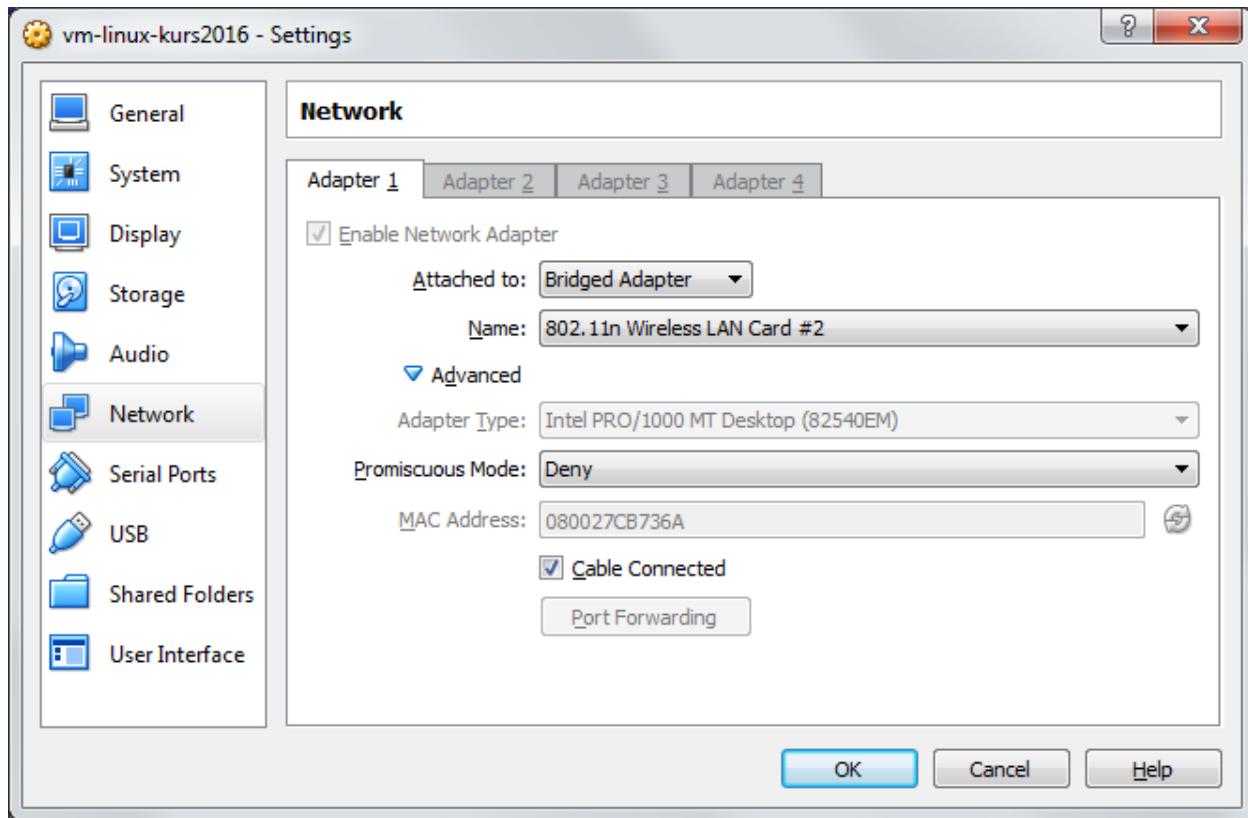
Слика 21 Подешавања нове мрежне конекције

За Netmask можете користити код 24, а поље Gateway можете оставити празно (уколико кликнете на поље Gateway, мораћете да укуцате валидну IP адресу иначе нећете моћи да сачувате подешавања кликом на Save дугме.

(Опција 2) Повезивање Raspberry Pi на локалну мрежу

Мрежним каблом, повежите Ethernet порт Raspberry Pi плочице са портом на етернет свичу, тј. на локалну мрежу. Уколико радите у виртуелној машини, потребно подесити мрежни адаптер као *Bridged Adapter* (Слика 22).

Линукс курс 2018/2019



Слика 22 Подесавање локалног мрежног адаптера на виртуелној машини

Наставак у U-boot командној линији

Вратите се на U-boot командну линију, поставите променљиве (ipaddr и serverip) у складу са дефинисаном адресом:

```
setenv ipaddr <IP адреса rpi>
```

```
setenv serverip <IP адреса pc>
```

Сачувате ове измене на SD картицу командом:

```
saveenv
```

Можете тестираТИ TFTP конекцију. Најпре направите малу текстуалну датотеку у /var/lib/tftpboot.

Након тога, из U-Boot-а покрените



Линукс курс 2018/2019

```
tftp 0x01000000 textfile.txt
```

tftp команда би требало да је пребацила датотеку textfile.txt са вашег рачунара на плочу, на меморијску локацију 0x01000000 (ова локација је део DRAM на плочи). Можете то и верификовати читањем садржаја меморије командом:

```
md 0x01000000
```

Уколико имате проблема са пребацивањем датотеке преко tftp:

- Проверите да ли са рачунара можете да прозовете Raspberry Pi:

```
ping <IP адреса rpi>
```

Може се десити да не успевате да га прозовете, али оставите покренут ping да видите да ли ће проћи у тренутку док радите наредни корак (разлог: мрежа на Raspberry Pi у U-boot-у није стално активна већ само по потреби)

- Проверите да ли са Raspberry Pi можете да прозовете рачунар:

```
ping <IP адреса pc>
```

ping \$serverip # овим уједно проверавате и да ли је варијабла serverip коректно подешена, обратите пажњу на IP адресу коју Raspberry Pi том приликом користи (променљива ipaddr)

- Проверите да ли се датотека коју покушавате да копирате налази на путањи:

```
ls /var/lib/tftpboot
```

- Покрените поново сервис tftpd-hpa:

```
sudo service tftpd-hpa restart
```

Прекопирајте датотеке zImage и bcm2709-rpi-2-b.dtb које су прекомпајлиране и налазе се на путањи linux-kernel-labs/bootloader/rpi-2-b/ у home директоријум tftp server-а (/var/lib/tftpboot), како би касније покретање Линукс језгра са те путање преко tftp протокола било могуће.

Сада смо спремни за покретање Линукс језгра!



Линукс курс 2018/2019

Подешавање NFS server-a

Инсталирајте NFS server инсталацијем пакета nfs-kernel-server. По инсталацији, измените /etc(exports датотеку као root корисник тако што ћете додати следеће линије:

```
/home/<user>/linux-kernel-labs/modules/nfsroot      <IP      адреса  
rpi>(rw,no_root_squash,no_subtree_check)
```

Обратите пажњу да су путања и опције у истој линији. Такође проверите да нема размака између IP адресе и NFS опција, иначе ће се користи подразумеване опције за ову адресу, те ће root filesystem бити read-only.

Након тога рестартујте NFS server:

```
sudo /etc/init.d/nfs-kernel-server restart
```

Ако постоји порука о грешци, то најчешће значи да постоји синтаксна грешка у /etc(exports датотеци. Исправите грешке пре наставка!

Уколико желите да проверите шта тренутно NFS server обухвата од путања и адреса, можете то прегледати са:

```
showmount -e localhost
```

Boot-овање система

Најпре покрените плочу до U-Boot промпта. Пре boot-овања језгра, морамо подесити која се конзола користи и који root filesystem ће бити mount-ован преко NFS-а, подешавањем одређених параметара језгра.

Урадите ово подешавањем U-boot bootargs променљиве (све у једној линији, обратите пажњу):

```
setenv bootargs "root=/dev/nfs    rw    ip=<IP      адреса      rpi>  
console=ttyAMA0,115200          nfsroot=<IP      адреса      pc>:/home/<user>/linux-kernel-labs/modules/nfsroot"  
  
saveenv
```

Ако касније желите да промените подешавања, променљиве можете мењати са:

```
editenv bootargs
```



Линукс курс 2018/2019

Сада пребаците image језгра кроз tftp:

```
tftp 0x01000000 zImage
```

Такође треба пребачити и device tree blob:

```
tftp 0x02000000 bcm2709-rpi-2-b.dtb
```

Сада можете покренути језгро:

```
bootz 0x01000000 - 0x02000000
```

Ако је све у реду, требало би да добијете login промпт (корисник: root, без лозинке). У супротном, проверите подешавања и питајте асистента за помоћ ако се заглавите. Ако језгро не успе да mount-ује NFS filesystem, обратите пажњу на поруке о грешкама у конзоли. Ако немате идеју, погледајте и NFS server поруке у /var/log/syslog.

Аутоматизација процеса boot-овања

Да бисте избегли куцање истих команда сваки пут када се укључи плоча, можете користити U-Boot bootcmd променљиву:

```
setenv bootcmd "tftp 0x01000000 zImage; tftp 0x02000000 bcm2709-rpi-2-b.dtb; bootz 0x01000000 - 0x02000000"
```

```
saveenv
```

Слободно ово можете променити по потреби. Нпр. уколико желите да се вратите на boot-овање са SD картице:

```
setenv bootcmd run mmc_boot
```

```
saveenv
```

На тај начин корисите већ подешене променљиве mmc_boot и друге, а уколико желите сами да подесите преузимање језгра са SD картице и root fs-а:

```
setenv bootargs "/dev/mmcblk0p2 rootwait console=ttyAMA0,115200"
```

```
setenv bootcmd "fatload mmc 0:1 0x01000000 zImage; fatload mmc 0:1 0x02000000 bcm2709-rpi-2-b.dtb; bootz 0x01000000 - 0x02000000"
```



Линукс курс 2018/2019

saveenv

Провера да ли је root filesystem коректно mount-ован преко NFS-а

На рачунару направите кратку текст датотеку у root директоријуму root filesystem-а припремљеним за Raspberry Pi:

```
gedit linux-kernel-labs/modules/nfsroot/root/test.txt
```

На Raspberry Pi рачунару би исту датореку требало да видите на путању /root/test.txt, нпр:

```
cat /root/test.txt
```



Линукс курс 2018/2019

Упознавање са Линукс кернелом и превођење кернела

Циљ

Научити како да се преузме изворни код кернела и да се примене закрпе. Упознавање са изворним кодом. Упознавање са конфигурисањем и превођењем кернела. Покретање кернела ће бити омогућено на Raspberry Pi платформи. На њему ће бити покренут минималистички кернел.

Исход

Након ове вежбе ћете моћи да:

- Преузмете изворни код са званичне локације
- Примените закрпе за кернел
- Истражите изворни код, прегледате датотеке, претражите заглавља или друге информације
- Конфигуришете, преведете и покренете кернел на Raspberry Pi платформи

ПРЕУЗМИТЕ ИЗВОРНИ КОД

Позиционирајте се у директоријум `~/linux-kernel-labs/src`.

Са `github` репозиторијума преузмите код помоћу одговарајуће `git` команде:

```
git clone -b rpi-4.4.y --single-branch https://github.com/raspberrypi/linux --depth=1
```

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` и затим направите нову грану `dan04` и позиционирајте се у исту:

```
git checkout -b dan04
```



Линукс курс 2018/2019

За каснији рад (комитовање измена) на GIT-у, конфигуришите своје податке:

```
git config --global user.email "ime.prezime@mail"
```

```
git config --global user.name "Ime Prezime"
```

ПРИМЕНИТЕ ЗАКРПЕ

Инсталирајте patch команду или преко графичке спреге или преко командне линије користећи команду:

```
sudo apt-get install patch
```

На преузети код примените закрпу:

```
~/linux-kernel-labs/modules/data/LinuksKurs.patch
```

Прегледајте закрпу са vi или gvim да разумете информације које се преносе у таквој датотеци. Како су описане додате или уклоњене датотеке?

УПОЗНАЈТЕ СЕ СА КОДОМ

Како корисник кернела Линукса, често ћете морати да нађете која датотека имплементира дату функцију. Стога, корисно је упозанти се са изворним кодом кернела прегледањем датотека.

1. Пронађите слику Линукс логоа у извнорном коду.
2. Пронађите ко одржава 3C505 мрежни драјвер.
3. Пронађите декларацију `platform_device_register()` функције.

Поред ручне претраге кода постоје и алати који нам у томе помажу. Испробајте претрагу помоћу LXR (Linux Cross Reference) на адреси <http://lxr.free-electrons.com> и изaberite верзију Линукса најближу оној коју користите.

Такође, користан алат у ситуацијама када приступ интернету није омогућен је cscope. Коришћењем cscope алата и LXR пронађите претходно ручно пронађене датотеке и информације и уочите предност коришћења алата за



Линукс курс 2018/2019

индексирање. Да би се извршило индексирање целог Линукс кода са cscope алатом потребно је покренути команду cscope -Rk. Додатне информације о коришћењу овог алата пронаћи покретањем команде cscope --help. Уколико је потребно претходно инсталирајте cscope са командом sudo apt-get install cscope.

ИНСТАЛАЦИЈА АЛАТА ЗА ПРЕВОЂЕЊЕ

Пре него што пређемо на конфигурисање и превођење Линукс језгра потребно је инсталирати скуп алата за превођење. За превођење Линукса за RPI плочу користићемо gcc-arm-linux-gnueabihf преводилац. Инсталирајте поменути преводилац са командом:

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

КОНФИГУРАЦИЈА КЕРНЕЛА

Пошто кернел преводимо за RPI плочу која подржава ARM скуп инструкција, а само превођење се обавља на x86 архитектури, најпре је потребно подесити одговарајуће окружење за превођење. Минимално што је потребно да се уради је да се подесе циљна архитектура и префикс алата за превођење који ће се користити. Ово се може урадити на више начина од којих сваки има своје предности и мане.

Први начин је директна промена вредности ARCH и CROSS_COMPILE варијабли у Makefile датотеци. Овај приступ је погодан уколико кернел увек преводимо за исту архитектуру и са истим преводиоцем. Ипак, како то није чест случај, овај начин подешавања окружења за превођење се не препоручује.

Други начин, који је и најчешћи је подешавање варијабли окружења ARCH и CROSS_COMPILE помоћу команде export. Уколико желимо да подесимо окружење за превођење за RPI плочу на овај начин потребно је да извршимо следеће команде:

```
export ARCH=arm
```

```
export CROSS_COMPILE=arm-linux-gnueabihf-
```



Линукс курс 2018/2019

Мана овог приступа је у томе што су подешене варијабле видљиве само у терминалу у ком су извршена претходно наведене команде.

Трећи начин подешавања окружења за превођење је да се вредности варијабли ARCH и CROSS_COMPILE проследе при сваком позиву make команде, на пример:

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j2
```

При коришћењу овог приступа постоји велика шанса да се у неком од позива make команде изоставе потребне варијабле што резултује грешком у превођењу.

Након што сте подесили окружење за превођење за RPI плочу можете да приступите конфигурању кернела.

Позиционирајте се у преузети и patch-овани кернел директоријум. Примените подразумевану конфигурацију која је додата у изворни код кернела применом закрпе. Искористите у те сврхе наредну команду која ће бити објашњена на наредним предавањима и вежбама:

```
make bcm2709_rtrk_linux_kurs_defconfig
```

Покрените команду xconfig да покренете спрегу за конфигурацију кернела. Спрега за конфигурацију кернела је дата као изворни код у самом кернелу и команда make xconfig је преводи аутоматски, али су неопходна заглавља и библиотеке. Можда ће вам бити потребно да инсталirate додатне пакете, попут libqt4-dev који садржи Qt развојне датотеке, као и g++ пакет који представља C++ преводилац.

У покренутој спрези за конфигурацију, укључите Options --- Show Name опцију. Ово је понекад корисно, када је име параметра експлицитније од описа или када пратимо нека упутства која су нам задата са именима параметара.

Такође, пробајте и опције Options --- Show All Options и Options --- Show Debug Info. Ове опције ће вам дозволити да видите све параметре који се иначе не би приказали, јер зависе од вредности других параметара. Тако што кликнете на опис неког од тих параметара, видећете његове предуслове за укључивање и разумећете зашто није могуће да се одаберу.



Линукс курс 2018/2019

Специфицирајте суфикс верзију; тако ћете моћи да разликујете свој кернел командом uname -r или cat /proc/version, кад покренете систем.

Конфигуришите мали кернел за RPI плочу:

- Broadcom BCM2709 Development platform (ARCH_BCM2709)
- Подршка за MMC
- Ext4 и vfat системе датотека
- Подршка за elf binaries (BINfmt_ELF)
- Подршка за NEON у кернел моду
- MMC DMA подршка за BCM2835 Arasan контролер
- SDHCI подршка за MMC
- SDHOST подршка за BCM2835
- Укључите ARM errata 643719
- Укључите подршку за ARM Architected timer
- **Искључите** подршку за L2X0 cache
- Поставите LOCALVERSION на вашу верзију

Одвојите време да прегледате нове одлике које укључујете!

Такође, можете пробати make menuconfig. Иако није графичка спрега, неки људи преферирају овај начин конфигурације. Пошто је menuconfig спрега базирана на ncurses библиотеки, мораћете да инсталирате libncurses-dev пакет да бисте је користили.

ПРЕВЕДИТЕ КЕРНЕЛ

Потребно је само да покренете:

```
make -j 4
```



Линукс курс 2018/2019

Добијени кернел је потребно покренути на RPI плочи. Датотеке које су добијене превођењем је потребно копирати да би их RPI преузимао преко мреже у процесу покретања.

Обзиром да у кернел није укључена подршка за NFS, U-Boot треба подесити тако да се `rfs` преузима са картице. Покрените из U-Boot-а нови кернел тако што ћете привремено (без позива `saveenv`) променити променљиву `bootargs`, а потом покренути кернел:

```
setenv bootargs root=/dev/mmcblk0p2 rootwait console=tty1  
console=ttyAMA0,115200
```

```
boot
```

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan04 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Конфигурисање и превођење кернела

Циљ

Упознавање са конфигурисањем и превођењем кернела.

Исход

Након ове вежбе ћете моћи да:

- Поставите и модификујете слику коренског система датотека тако што ћете додавати уносе у /dev/ директоријум

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` који садржи кернел за RPI са github репозиторијума <https://github.com/raspberrypi/linux> и затим од полазне гране `rpi-4.4.y` направите нову грану `dan05` и позиционирајте се у исту:

```
git checkout rpi-4.4.y  
git checkout -b dan05
```

Позиционирајте се у директоријум `~/linux-kernel-labs/modules` који садржи коренски систем датотека за RPI и затим од полазне гране `master` направите нову грану `dan05` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan05
```

Распакујте у тренутни директоријум (`~/linux-kernel-labs/modules`) архиву са коренским системом датотека посебно припремљену за ову вежбу:

```
sudo tar xvjf nfsrootVezba.tar.bz2
```



Линукс курс 2018/2019

Поставите на следећи начин власнике поједињих датотека:

```
sudo chown -R rtrk.rtrk nfsrootVezba/root
```

КОНФИГУРАЦИЈА КЕРНЕЛА

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` и искористите дату конфигурацију `~/linux-kernel-labs/modules/data/Linux.config` као тренутну (`.config`). Покрените команду `make xconfig` да покренете спрегу за конфигурацију кернела.

Конфигуришите кернел за употребу `direct frame buffer`-а и његове конзоле, као и `frame buffer` подршку за `BCM2708` платформу (`FB_BCM2708`). Поставите свој суфикс на верзију кернела.

Одвојите време да прегледате нове одлике које укључујете!

Напомена: Уколико у радном директоријуму линукс језгра који желимо да конфигуришемо (`~/linux-kernel-labs/src/linux`) већ постоји нека претходна конфигурација (`.config`) онда је пре новог подешавања препоручено извршити команду `'make distclean'` како не би дошло до мешања датотека и евентуално појаве грешке при превођењу и увезивању коначне верзије кернел језгра. Такође, не заборавите да подесите `ARCH` и `CROSS_COMPILE` варијабле.

ПРЕВЕДИТЕ КЕРНЕЛ

Потребно је само да покренете:

```
make -j 4
```

По потреби, уколико је верзија кернела новија од оне коришћене у `nfsroot` коренском систему датотека, додајте нове модуле командом:

```
make modules_install INSTALL_MOD_PATH=<путanja до nfsroot директоријума>
```



Линукс курс 2018/2019

ПОКРЕТАЊЕ КЕРНЕЛА СА КОRENСКИМ СИСТЕМОМ ДАТОТЕКА ПРЕКО МРЕЖЕ (ПО ПОТРЕБИ)

Инсталирајте NFS сервер: `nfs-kernel-server` пакет. Када је исконтиран, измените `/etc/exports` датотеку као `root` и додајте путању до вашег коренског система датотека, као и IP путању до RPI плоче.

Након тога, рестартујте NFS сервер:

```
sudo /etc/init.d/nfs-kernel-server restart
```

По потреби подесите путању до директоријума са коренским системом датотека `y bootargs y u-boot-y`.

Покрените кернел.

ДОДАВАЊЕ УРЕЂАЈА У КОRENСКУ ДАТОТЕКУ СИСТЕМА

Позиционирајте се у ваш мрежни коренски систем датотека.

Креирајте `ttyAMA0` датотеку карактерног уређаја са бројевима 204 и 64.

Уколико је потребно, додајте још датотека уређаја.

Сада креирајте `dev/console` уређај који недостаје. Можете проверити `/dev/console` датотеку уређаја на радној машини да пронађете тип датотеке, као и велике и мале бројеве.

Поново покрените RPI плочу. Проверите да ли постоји `console y dev` директоријуму који сте креирали. Покрените неколико команди.

РОТИРАЊЕ КОНЗОЛЕ



Линукс курс 2018/2019

Укључите у конфигурацији опцију за ротирање конзоле и додајте ротацију у аргументе који се прослеђују кернелу. Више информација пронађите у Documentation/fb/fbcon.txt

Поново покрените RPI са новим изменама.

АНИМАЦИЈЕ

Креирајте fb0 датотеку карактерног уређаја са бројевима 29 и 0 на RPI плочи.

Ако сте исправно додали опције за direct framebuffer подршку, моћи ћете, после додавања потребних датотека уређаја, да покренете анимације на RPI плочи:

```
df_andi  
df_texture
```

Има и осталих, потражите их на плочи!

ВЕРЗИЈА КЕРНЕЛА

Проверите верзију кернела и будите сигурни да користите вашу верзију кернела тако што ћете очитати суфикс са верзије кернела.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. ~/linux-kernel-labs и исто за ~/linux-kernel-labs/src/linux:

```
git add -A  
git commit -as -m "dan05 zavrsen"
```



Линукс курс 2018/2019

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Прављење и покретање уgraђеног система

ЦИЉ

Савладати прављење и покретање малог уgraђеног система.

ИСХОД

Након ове вежбе ћете моћи да:

- Преузмете, преведете и инсталирате busybox
- Развијете мали компелтан rfs
- Генеришете initramfs

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src` и затим од полазне гране `master` направите нову грану `dan06` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan06
```

Преузмите изворни код busybox-а командом:

```
wget http://www.busybox.net/downloads/busybox-1.21.1.tar.bz2
```

и распакујте добијену архиву.

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` који садржи кернел за RPI са github репозиторијума <https://github.com/raspberrypi/linux> и затим од полазне гране `rpi-4.4.y` направите нову грану `dan06` и позиционирајте се у исту:

```
git checkout rpi-4.4.y
```



Линукс курс 2018/2019

git checkout -b dan06

КОНФИГУРАЦИЈА И ПРЕВОЂЕЊЕ ЛИНУКС КЕРНЕЛА

За конфигурацију Линукса **искористите конфигурациону датотеку** `~/linux-kernel-labs/modules/data/tinyLinux.config` и преведите Линукс кернел.

Напомена: Не заборавите да подесите ARCH и CROSS_COMPILE варијабле.

КОНФИГУРАЦИЈА И ПРЕВОЂЕЊЕ BUSYBOX-А

Позиционирајте се у директоријум у који сте распаковали busybox. За конфигурацију busybox-а **искористите конфигурациону датотеку** `~/linux-kernel-labs/modules/data/tinyBusybox.config`. Додатно, укључите опцију да се busybox преведе са статички увезаним библиотекама и подесите путању на коју ће се инсталеријати. Преведите и инсталеријате busybox у празан директоријум. Покрените претходно преведени Линукс кернел. За rfs користите директоријум у који је инсталариран busybox, преко мреже (NFS). Додајте све потребне датотеке уређаја да би се добила функционална Линукс конзола.

ДОДАВАЊЕ PROC И SYS СИСТЕМА ДАТОТЕКА

Направите директоријуме `/etc/init.d`, `/proc` и `/sys` у оквиру rfs-a. На proc директоријум маунтовати proc систем датотека, а на sys директоријум sysfs. Аутоматизовати моунтовање ових система датотека на старту система тако што ћете направити и на адекватан начин попунити датотеку `/etc/init.d/rcS`. То је скрипта коју треба да покрене први кориснички програм (скрипта), `/etc/inittab`, који такође треба направити и попунити са:

```
::sysinit:/etc/init.d/rcS
```

```
ttyAMA0::askfirst:/bin/sh
```



Линукс курс 2018/2019

ИМПЛЕМЕНТАЦИЈА WEB СПРЕГЕ ЗА ВАШ УРЕЂАЈ

Копирати директоријум `~/linux-kernel-labs/modules/data/www`, тако да буде у коренском директоријуму (`/www`) rfs-а. Покренути busybox web послужиоц из командне линије:

```
/usr/sbin/httpd -h /www/ &
```

Тестирали да ли ваша web спрега ради тако што ћете на host-у, односно на вашој радној станици отворити:

```
http://<адреса rpi>
```

Поправити позиве ка `/proc` унутар web спреге. Аутоматизовати покретање web послужитеља на старту система додавањем одговарајуће команде у `rcS` датотеку.

ПРЕБАЦИВАЊЕ НА ДЕЉЕНЕ БИБЛИОТЕКЕ (SHARED LIBRARIES)

Креирајте мали Це програм (`hello world`) који ћете превести за RPI платформу, динамички увезан са библиотекама, и покрените га на платформи.

Поправите грешке копирањем потребних дељених библиотека из `toolchain-a` у `lib/` директоријум циљне платформе док програм не проради.

Савет: почети са копирањем `ld-linux-armhf.so.3` – динамички увезивач потребан за покретање било ког не-статички преведеног програма. Користити `readelf` из алата за превођење за откривање потребних библиотека.

Када тестни програм проради, поново преведите busybox без опције за статичко превођење, тако да busybox искористи предности дељених библиотека које се сада налазе на циљној платформи.

ГЕНЕРИСАЊЕ INITRAMFS

Искористите генерисани rfs као изврнне датотеке за генерисање initramfs-а (`CONFIG_INITRAMFS_SOURCE`) и поново преведите кернел. Претходно је



Линукс курс 2018/2019

потребно преименовати `linuxrc` у `init`, у коренском директоријуму `rfs`-а. Покрените `initramfs` на RPI. Потребно је изменити `bootargs` варијаблу у `u-boot`-у тако да у њој остану само параметри `rw` и `console`.

По покретању `initramfs`-а додајте све што је потребно да се оствари иста функционалност као и када је `rfs` био на мрежи.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan06 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Писање и превођење модула

ЦИЉ

Научити како се пишу и преводе модули.

ИСХОД

Након ове вежбе ћете моћи да:

- Напишете кернел модул са неколико могућности, укључујући параметре
- Приступите кернелу из вашег модула
- Поставите окружење за превођење модула
- Додате код вашег модула у кернел стабла и да изградите кернел закрпу из вашег новог извornog кода
- Креирате једноставан знаковни руковалац
- Добијете од кернела слободан major број и направите одговарајућу датотеку уређаја
- Коришћењем kmalloc и kfree функција управљате меморијом.

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` који садржи кернел за RPI са github репозиторијума <https://github.com/raspberrypi/linux> и затим од полазне гране `rpi-4.4.y` направите нову грану `dan08` и позиционирајте се у исту:

```
git checkout rpi-4.4.y
```

```
git checkout -b dan08
```



Линукс курс 2018/2019

Позиционирајте се у директоријум `~/linux-kernel-labs/modules` који садржи коренски систем датотека за RPI и затим од полазне гране `master` направите нову грану `dan08` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan08
```

КОНФИГУРАЦИЈА И ПОКРЕТАЊЕ ЛИНУКС КЕРНЕЛА

За конфигурацију је потребно искористити подразумевану конфигурацију за `bcm2709` (`bcm2709_defconfig`). Пре постављања конфигурације и покретања превођења кернела потребно је подесити одговарајуће окружење за унакрсно превођење (`ARCH` и `CROSS_COMPILE`). Након примене подразумеване конфигурације додатно подесити сопствени суфикс за верзију кернела. Преведен кернел покренути из `U-boot-a`. За `rfs` користити директоријум `nfsroot` преко мреже.

ПИСАЊЕ И ПРЕВОЂЕЊЕ МОДУЛА

Позиционирати се у `nfsroot/root/hello` директоријум. Попунити `hello_version.c` датотеку са одговарајућим кодом тако да се при учитавању модула прикаже порука:

```
Hello Master. You are currently using Linux <version>.
```

Такође, потребно је приказати и поздравну поруку приликом уклањања модула.

У истом директоријуму се налази и `Makefile` датотека која омогућава превођење модула ван структуре кернел стабла. Упознајте се са садржајем ове датотеке, по потреби прилагодите путање и преведите модул.

Напомена: Обавезно је поставити окружење за унакрсно превођење и приликом превођења модула.



Линукс курс 2018/2019

ТЕСТИРАЊЕ МОДУЛА – САВЕТИ

Учитати нови модул. Проверити да ли ради као што је очекивано. Док не проради, уклонити га, модификовати код, превести и поново покренути колико год је пута потребно.

Покренути команду за проверу учитаних модула. Након тога, покушати доћи до исте информације само коришћењем `cat` команде.

ДОДАВАЊЕ ПАРАМЕТАРА У МОДУЛ

Додати “`who`” параметар у модул. Модул би требало да испише “Hello <`who`>...” уместо “Hello Master...”.

Превести и тестирати модул проверавајући да ли прихвата `who` параметар при учитавању.

ДОДАВАЊЕ ИНФОРМАЦИЈЕ О ВРЕМЕНУ

Изменити поздравну поруку модула тако да прикаже и колико је секунди протекло од учитавања модула.

За добављање тренутног времена се може користити функција `do_gettimeofday()`.

ЈЕДНОСТАВАН ЗНАКОВНИ РУКОВАЛАЦ

Претворити модул у једноставан знаковни руковалац. Препустити кернелу доделу `major` броја, а за `minor` број користити 0. Попунити `read`, `write` и `ioctl` операције модула на следећи начин:

`read`

Попунити бафер из корисничког простора садржајем бафера динамички заузетог приликом иницијализације модула.



Линукс курс 2018/2019

write

Потребно је прослеђени стринг из корисничког простора сместити у унапред динамички заузет бафер.

ioctl

У зависности од прослеђене команде урадити следеће:

- 0 – пребацити у мала слова цео садржај бафера динамички заузетог приликом иницијализације модула
- 1 - пребацити у велика слова цео садржај бафера динамички заузетог приликом иницијализације модула

Омогућити приступ модулу из корисничког простора креирањем одговарајуће датотеке уређаја уколико је потребно тестирати и додате функције. За тестирање *write* операције користити команду *echo*, за *read* операцију користити команду *cat*, а за *ioctl* операцију превести *ioctl.c* са истим преводиоцем и обавезно **статичким уvezивањем** (опција *-static*) и користити добијени програм као тестни.

ЛИНУКС СТАНДАРД КОДОВАЊА

Приликом кодовања требало би да се придржавате стриктних стандарда кодовања, уколико желите да једног дана и тај код буде укључен у изворни код Линукса. Један од главних разлога за то је читљивост кода. Ако би свако користио свој стил, узимајући у обзир велики број контрибутора, читање кода кернела би било веома непријатно.

Срећом, Линукс кернел заједница нуди алат за проверу доследности у придржавању стандарду кодовања.

Покрените *scripts/checkpatch.pl -h* команду у извornом коду кернела да бисте проверили доступне опције. Након тога покрените:

```
~/linux-kernel-labs/src/linux/scripts/checkpatch.pl --file  
--no-tree hello_version.c
```



Линукс курс 2018/2019

Проверите колико недоследности је пријављено за ваш код и уносите исправке док год има недоследности. Уколико је пријављено много грешака, проверите да ли вам је на коректан начин конфигурисан едитор, према правилима стила кодовања кернела из Documentation/CodingStyle.

ДОДАВАЊЕ HELLO_VERSION КОДА У КЕРНЕЛ КОД

Додати изворни код модула у `drivers/misc/` директоријум у кернел коду. Модификовати одговарајуће конфигурационе датотеке и Makefile како би модул постао видљив у конфигурационим алатима (`xconfig`, `menuconfig...`) и како би могао да се преведе заједно са кернелом.

Када модул постане видљив у конфигурационим алатима изабрати да се преводи као модул. Покренути превођење кернела. По завршетку превођења кернела, модуле инсталирати у `nfsroot` командама:

```
sudo chown -R rtrk.rtrk <путања> do nfsroot  
direktorijuma>/lib/  
  
make modules_install INSTALL_MOD_PATH=<путања> do nfsroot  
direktorijuma>  
  
sudo chown -R root.root <путања> do nfsroot  
direktorijuma>/lib/
```

КРЕИРАЊЕ ЗАКРПЕ

Креирати закрпу која ће додавати изворни код новог модула у чист кернел код, али и извршити потребне промене у конфигурационим датотекама и Makefile-овима.

Тестирати закрпу применом на чист кернел код.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од



Линукс курс 2018/2019

директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan08 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Дебаговање руковалаца и кернела

ЦИЉ

Научити како се користи debugFs и како се проналазе грешке у кернел коду.

ИСХОД

Након ове вежбе ћете моћи да:

- Креирате debugFs уносе у оквиру свог модула
- Пронађете грешку у оквиру кернел кода
- Декомпајлирате кернел

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src` и затим од полазне гране `master` направите нову грану `dan08` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan08
```

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` који садржи кернел за RPI са github репозиторијума <https://github.com/raspberrypi/linux> и затим од полазне гране `rpi-4.4.y` направите нову грану `dan08` и позиционирајте се у исту:

```
git checkout rpi-4.4.y  
git checkout -b dan08
```



Линукс курс 2018/2019

PR_DEBUG() И ДИНАМИЧКО ДЕБАГОВАЊЕ

Додајте `pr_debug()` исписе у све функције `hello_version` модула. Такође, додајте и глобалне бројаче који прате укупан број уписаних бајта, укупан број прочитаних бајта и тренутно доступан број бајта за читање.

Проверите шта се дешава са модулом? Да ли су исписи који су додати видљиви? Уколико нису видљиви, укључите `CONFIG_DYNAMIC_DEBUG` и поново преведите кернел. Након превођења маунтујте `debugfs` и конфигуришите исписе на следећи начин:

1. Излистајте све доступне дебаг поруке у кернелу,
2. Укључите све поруке из `hello_version` модула и проверити да ли су заиста видљиве,
3. Укључите само једну поруку из `hello_version` модула и уверите се да се види само та порука, а остале не.

Сада имате добар механизам да држите пуно дебаг исписа у модулу и селективно бирате које ћете да прикажете.

DEBUGFS

Користећи `debugfs` додајте нове уносе у дебаг систему датотека. Додајте:

- Директоријум `hello` у оквиру `debugfs` система датотека,
- У том директоријуму датотеке које ће да приказују вредности глобалних бројача.

Поново преведите и учитајте модул и проверите да ли су новокреирани уноси у оквиру `debugfs`-а видљиви на путањи `/sys/kernel/debug/hello/`.



Линукс курс 2018/2019

АНАЛИЗА ГРЕШКЕ У КЕРНЕЛУ

Позиционирајте се у директоријуму `linux-kernel-labs/modules/nfsroot/root/debugging/`.

Проверите да ли је кернел преведен са:

- укљученим `CONFIG_DEBUG_INFO`
- искљученим `CONFIG_ARM_UNWIND`

Преведите модул `drvbroken`, покрените га на платформи и видите поруку о грешци.

АНАЛИЗА ПОРУКЕ О ГРЕШЦИ

Пажљиво анализирајте поруку о грешци. Знајући да на ARM-у РС регистар садржи локацију инструкције која се извршава, покушајте да откријете која функција је изазвала грешку у кернелу.

Коришћењем LXR-а кернел кода, погледати декларацију проблематичне функције. Ово би требало да буде доволно да се открије и отклони грешка.

ДУБЉА АНАЛИЗА ПРОБЛЕМА

У случају када се једноставним прегледом кода и поруке о грешци не може утврдити шта је проблем, може да се погледа дисасемблирана верзија функције на један од два начина:

1.

```
cd linux-kernel-labs/src/linux
arm-linux-gnueabihf-objdump -S vmlinux > vmlinu.x.disasm
```
2. коришћењем `gdb-multiarch`

```
sudo apt-get install gdb-multiarch
gdb-multiarch vmlinux
```



Линукс курс 2018/2019

```
(gdb) set arch arm  
(gdb) set gnutarget elf32-littlearm  
(gdb) disassemble function_name
```

Даље је могуће пронаћи тачну инструкцију на којој је дошло до грешке. Померај унутар функције се може видети у поруци о грешци, а такође се може видети и део кода око проблематичне инструкције.

Наравно, за овакву врсту дебаговања је потребно одређено знање асемблера за дату архитектуру.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan08 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Упознавање са buildroot алатом и U-boot-ом

ЦИЉ

Научити како да користите buildroot и U-boot.

ИСХОД

Након ове вежбе ћете моћи да:

- Генеришете toolchain са buildroot-ом
- Генеришете комплетан rfs и кернел са buildroot-ом
- Запишете кернел и rfs на трајну меморију

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src` и затим од полазне гране `master` направите нову грану `dan09` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan09
```

Напомена: За buildroot је неопходно инсталацији пакет `device-tree-compiler`.

Преузмите изворни код buildroot-а следећим командама на путању `~/linux-kernel-labs/src/`:

```
git clone git://git.buildroot.net/buildroot (клонира ГИТ репозиторијум)  
cd buildroot  
git checkout 476067a (преузети одређену верзију са репозиторијума)
```



Линукс курс 2018/2019

Преименујте добијени директоријум са buildroot-ом (нпр. у buildroot_tc, за генерисање toolchain-а), па поновите исту процедуру за генерисање комплетног rfs-а са кернелом (преименовати нпр. у buildroot_rfs).

Напомена: Нема потребе поново клонирати гит репозиторијум, довољно је копирати садржај buildroot директоријума у нове buildroot_tc и buildroot_rfs директоријуме.

Преузмите изворни код U-boot-а са гит репозиторијума <git://git.denx.de/u-boot.git> на путању `~/linux-kernel-labs/src/`. Након преузимања репозиторијума позиционирати се на одређену верзију:

```
cd u-boot  
git checkout a705ebc
```

ГЕНЕРИСАЊЕ TOOLCHAIN-А

Почните од минималистичке конфигурације buildroot-а (`make allnoconfig`). Који директоријум користимо, buildroot_tc?

Коришћењем `xconfig` алата, изаберите следећу конфигурацију:

- Target options
 - target Architecture -> ARM(little endian)
 - Target Binary Format (ELF)
 - Target Architecture Variant (cortex-A7)
 - Target ABI (EABIhf)
 - Floating point strategy (NEON/VFPv4)
 - ARM instruction set (ARM)
- Build options
 - Enable compiler cache
- Toolchain



Линукс курс 2018/2019

- Toolchain type (Buildroot toolchain)
- C library (uClibc/ng)
- Kernel Headers (Linux 4.8.x kernel headers)
- Enable RPC support
- Enable WCHAR support
- Enable toolchain locale
- Thread library debugging
- Enable stack protection
- Compile and install uClibc utilities
- Enable C++ support
- Build cross gdb for host
- Enable MMU support
- System configuration
 - Init system (None)
 - Purge unwanted locales
- Target packages
 - **Искључити** Busybox

Покренути превођење са командом `make -j4` или `make toolchain -j4`. Превођење можете покренути у пару са колегом поред себе па на крају продискутовати време превођења у оба случаја. Резултат превођења ће се налазити на путањи `output/host`. По завршетку превођења генерисани toolchain тестирајте превођењем једноставног Hello world програма и његовим покретањем на циљној платформи.

Напомена: Како се toolchain не налази на некој од уобичајених путања, потребно је експортовати `PATH` и `LD_LIBRARY_PATH` варијабле на следећи начин:

```
export PATH=<путања до toolchain-a>/bin:$PATH
```

CONFIDENTIAL under NDA - Reproduction prohibited without the prior permission of RT-RK



Линукс курс 2018/2019

```
export LD_LIBRARY_PATH=<путања do toolchain-
a>/lib:$LD_LIBRARY_PATH
```

ГЕНЕРИСАЊЕ КОМПЛЕТНОГ RFS-А СА КЕРНЕЛОМ

У овом кораку треба искористити закрпу која се налази на путањи `~/linux-kernel-labs/bootloader/rpi-2-b/src/0001-RPI2-customization-for-RTRK-EMBEDDED-LINUX-PROGRAMMI.patch`. Након позиционирања у `buildroot` директоријум за генерисање RFS-а (`buildroot_rfs?`), за примену закрпе искористите следећу команду,:

```
git apply <путања до zakrpe>
```

Након примене закрпе примените подразумевану конфигурацију за RPI командом `make raspberrypi2_defconfig`. Коришћењем `xconfig` алата прегледајте које су опције укључене у подразумеваној конфигурацији. Након тога покрените превођење.

КОНФИГУРАЦИЈА И ПРЕВОЂЕЊЕ U-BOOT-А

Пре постављања конфигурације и покретања превођења кернела је потребно подесити одговарајуће окружење за унакрсно превођење (`ARCH` и `CROSS_COMPILE`). За конфигурацију U-boot-а искористите подразумевану конфигурацију за RPI платформу (`rpi_2_defconfig`). Коришћењем `xconfig` алата проверите да ли је укључена подршка за команде `ext` и `fat` система датотека и, ако није, укључите је. Такође, додајте суфикс верзије (`LOCALVERSION`) како би се при покретању добила потврда да је покренута нова верзија U-boot-а. Покрените превођење командом `make all -j4`. Резултат превођења `u-boot.bin` пребаците у коренски директоријум `tftp` послужиоца.

ЗАМЕНА U-BOOT-А НА МЕМОРИЈСКОЈ КАРТИЦИ

Пronађите на којој партицији се налази `u-boot.bin` на меморијској картици. За излиставање свих партиција користите команду `mmc part`, а за



Линукс курс 2018/2019

преглед садржаја партиције `ext4ls` или `fatls` у зависности од система датотека који се налази на датој партицији.

Када је постојећи `u-boot.bin` пронађен, потребно га је заменити новом верзијом. Учитајте нову верзију `u-boot.bin`-а у меморију на произвољну адресу (нпр. `0x1000000`). Запишите `u-boot.bin` из меморије на SD картицу одговарајућом функцијом (`ext4write` или `fatwrite`). После овог корака рестартујте RPI и уверите се да се покреће нова верзија U-boot-а.

ЗАМЕНА КЕРНЕЛА И RFS-А НА МЕМОРИЈСКОЈ КАРТИЦИ

Након завршетка генерисања комплетног rfs-а са кернелом, генерисане резултате упишите на меморијску картицу и то:

- Запишите `zImage` и `.dtb` датотеке на исту партицију и на исти начин као и `u-boot.bin`
- Препишите комплетну партицију која садржи rfs на картици командом `mmc write` са генерисаном сликом rfs-а `rootfs.ext2` (**претходно се добро упознајте са коришћењем `mmc write` команде, јер погрешно коришћење може да доведе до губитка читаве партиције. Обратите пажњу на величину блока коју уз остале детаље можете сазнати командом `help mmc` имајући у виду да се сви параметри, поред адреса, команди `mmc write` задају у броју блокова у хексадецималном формату.**)

Покрените из U-boot-а нови кернел и rfs са меморијске картице командом `run mmc_boot`.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у директоријум репозиторијума који је мењан, `~/linux-kernel-labs`:

```
git add -A
```



Линукс курс 2018/2019

```
git commit -as -m "dan09 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Модел уређаја – I²C уређај

ЦИЉ

Савладати декларисање I²C уређаја и основних функција руковаоца које се позивају када је уређај детектован.

ИСХОД

Кроз наредне вежбе имплементираћемо руковалац једног I²C уређаја, који омогућава функционалност I²C nunchuk-а.

Након ове вежбе ћете моћи да:

- Додајете I²C уређаје (и сличне) у стабло уређаја (device tree).
- Имплементирате основне функције руковаоца `probe()` и `remove()`, те да се уверите да су поменуте функције позване када постоји поклапање доступног уређаја и руковаоца.
- Пронађете ваш руковалац и уређај на путањи `/sys`.

ПОСТАВКА

Позиционирајте се у директоријум `~/linux-kernel-labs/src` и затим од полазне гране `master` направите нову грану `dan10` и позиционирајте се у исту:

```
git checkout master  
git checkout -b dan10
```

Позиционирајте се у директоријум `~/linux-kernel-labs/src/linux` који садржи кернел за RPI са `github` репозиторијума <https://github.com/raspberrypi/linux> и затим од полазне гране `rpi-4.4.y` направите нову грану `dan10` и позиционирајте се у исту:

```
git checkout rpi-4.4.y
```

Линукс курс 2018/2019

git checkout -b dan10

ПОВЕЗИВАЊЕ NUNCHUK-А

Узмите nunchuk уређај који сте добили од асистента.

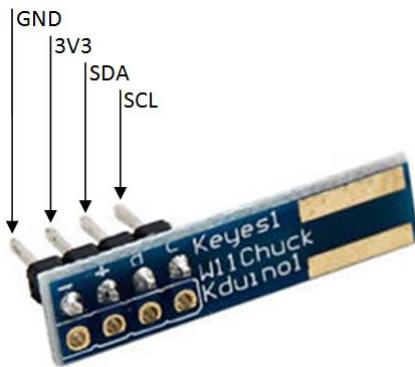
Повезаћемо га на други I²C пролаз процесора (i2c1), чији су пинови доступни на конектору J8.

Документ са корисним детаљима о nunchuk-у и конекторима можете преузети са адресе:

<http://free-electrons.com/labs/doc/nunchuk.pdf>

<http://www.robotshop.com/media/files/PDF/inex-zx-nunchuck-datasheet.pdf>

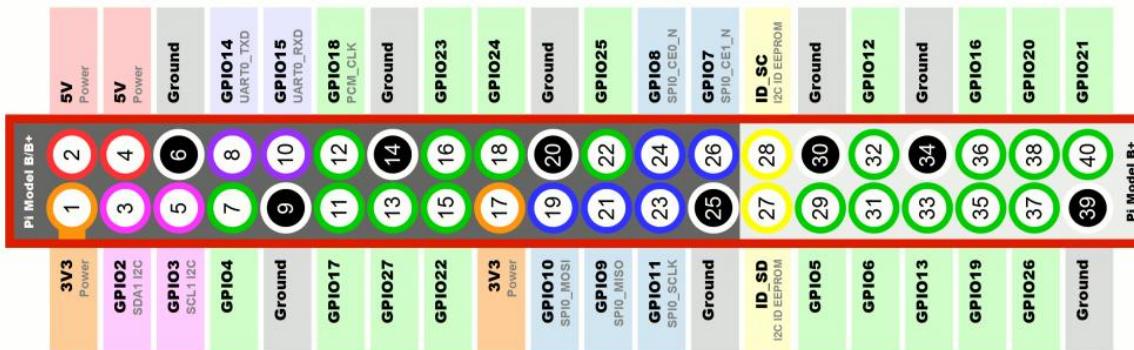
Сада можемо да идентификујемо 4 пина на nunchuk конектору:



Слика 1 Распоред пинова на конектору Nunchuk I²C уређаја

Отворите документ BCM2835 ARM Peripherals (<https://github.com/raspberrypi/documentation/blob/master/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf>) и пронађите SDA и SCL линије I2C1 периферије (SDA1 и SCL1) у табели 6-31 (GPIO Pins Alternative Function Assignment) где ћете пронаћи које се GPIO линије могу искористити у ове сврхе, а на слици испод видећете и који су то пинови на конектору J8.

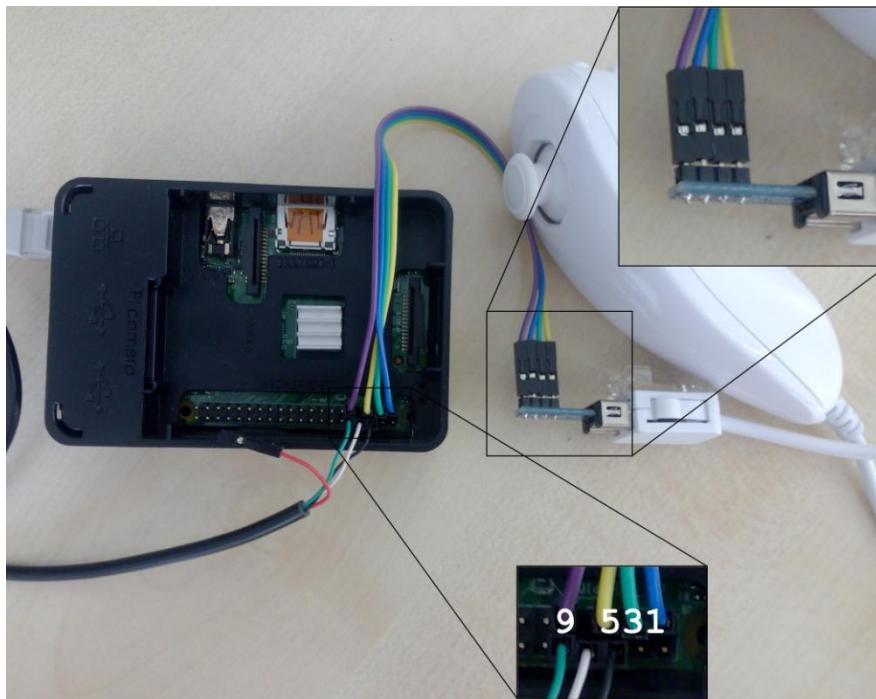
Линукс курс 2018/2019



Слика 2 Распоред пинова конектора J8 на Raspberry Pi 2 плочи

Сада повежите nunchuk конектор и Raspberry Pi 2 плочу на следећи начин, а исто је приказано и на слици испод:

- GND пин на J8 пин 9 (GND)
- 3V3 пин на J8 пин 1 (3V3)
- SCL пин на J8 пин 5 (GPIO3 / SCL1 I2C)
- SDA пин на J8 пин 3 (GPIO2 / SDA1 I2C)



Слика 3 Повезивање Nunchuk I²C уређаја и Raspberry Pi 2 плоче



Линукс курс 2018/2019

ПРАВЉЕЊЕ ПОСЕБНОГ СТАБЛА УРЕЂАЈА (DEVICE TREE)

Да бисмо Линукс језгру омогућили да рукује новим уређајем, морамо додати опис овог уређаја у стабло уређаја плоче (board device tree).

Како је стабло уређаја за Raspberry Pi 2 већ укључено у кернел и наставља да се развија за себе, ми нећемо правити измене директно на датотеци која се користи за ову плочу. Најбоље решење је да направимо посебно стабло уређаја за нашу плочу са нашим уређајем. На крају крајева, плочу са додатим било којим уређајем можемо посматрати као посебну плочу.

Дакле, наслеђивањем кода од Raspberry Pi 2 DTS, направите `bcm2709-rpi-2-b-custom.dts` датотеку. За сада из постојеће датотеке можете задржати само DTS `include` директиве, дефиницију модела, `compatible` подешавање, а од дефиниције уређаја задржите само оно што се тиче `uart-a` и `core-a`. Све остало можете обрисати.

Сада модификујте одговарајући Makefile да бисте били сигурни да ће се нова DTS датотека компајлирати аутоматски.

УКЉУЧИВАЊЕ ДРУГЕ I²C МАГИСТРАЛЕ

Пре свега требада омогућимо и конфигуришемо другу I²C магистралу (i2c1). Најпре пронађите DT дефиниције за i2c1.

Која је базна адреса њихових регистара? Пронађите исту адресу у документу BCM2835 ARM Peripherals¹.

Потом направите референцу на ову дефиницију у вашој DTS датотеци и омогућите ову магистралу. Такође, конфигуришите је да ради на 100 KHz. То је доволјно за сада.

¹ Претражујте I²C, али адресу периферије односно I²C контролера ћете пронаћи под називом Broadcom Serial Controller (BSC), па је други контролер у документу означен са BSC1.



Линукс курс 2018/2019

ДЕКЛАРАЦИЈА NUNCHUK УРЕЂАЈА У СТАБЛУ

Декларишите nunchuk уређај као чврт потомак i2c1 магистрале, одабирајући nintendo, nunchuk за compatible подешавање. I²C slave адресу nunchuk уређаја можете проверити у поменутом nunchuk документу².

ПРОВЕРА СТАБЛА УРЕЂАЈА У ПОКРЕНУТОМ СИСТЕМУ

Сада само преведите ваш DTB тражећи од Makefile језгра да поново преведе само DTB датотеке.

```
make dtbs
```

Ископирајте нову DTB датотеку на home директоријум tftp server-а (/var/lib/tftpboot/) и промените име DTB датотеке у U-Boot конфигурацији³ и покрените поново Raspberry Pi плочу.

Кроз директоријум /proc/device-tree можемо проверити Device Tree подешавања која је наш систем учитао. То је корисно када нисмо потпуно сигурни која су подешавања заправо учитана.

Нпр, можемо проверити присуство nunchuk чвора у стаблу уређаја:

```
# find /proc/device-tree/ -name "*nunchuk*"  
/proc/device-tree/soc/i2c@7e804000/nunchuk@52
```

Неко од подешавања уређаја након тога можете прочитати помоћу

```
cat /proc/device-tree/soc/i2c@7e804000/nunchuk@52/status
```

Користећи Device Tree Compiler (dtc, који је додат у root filesystem), можете такође проверити целу Device Tree структуру. То је бољи начин од провере изворних и include датотека у source директоријуму!

```
# dtc -I fs /proc/device-tree/
```

² Ова I²C slave адреса је одређена самим уређајем. Не можете је променити.

³ Савет: треба само да покренете editenv bootcmd, направите измене и сачувате са saveenv.



Линукс курс 2018/2019

Потражите `i2c1` и `nunchuk` у излазу након ове команде и уочите где су чворови инстанцирани.

ИМПЛЕМЕНТАЦИЈА ОСНОВНОГ I²C РУКОВАОЦА ЗА NUNCHUK

Најпре је потребно инсталирати модуле преведеног кернела у `nfsroot` командама:

```
sudo chown -R rtrk.rtrk <путанја> do nfsroot  
direktorijuma>/lib/
```

```
make modules_install INSTALL_MOD_PATH=<путанја> do nfsroot  
direktorijuma>
```

```
sudo chown -R root.root <путанја> do nfsroot  
direktorijuma>/lib/
```

Сада можемо кренути са писањем првих градивних блокова I²C руковаоца за наш `nunchuk` уређај.

У новом терминалу се позиционирајте на путању `~/linux-kernel-labs/modules/nfsroot/root/nunchuk/`. Овај директоријум садржи `Makefile` и скоро празну `nunchuk.c` датотеку.

Можете превести модул командом `make` (не заборавите да предходно подесите променљиве `ARCH` и `CROSS_COMPILE`). Како је тренутни директоријум део NFS root-а који се boot-ује на плочу, добијена `.ko` датотека ће одмах бити видљива и на плочи.

Ослањајући се на објашњења добијена у току предавања, попуните `nunchuk.c` датотеку да бисте имплементирали:

- `probe()` и `remove()` функције које ће бити позване када је `nunchuk` пронађен. За сада само искористите позив `pr_info()` унутар функција да бисте потврдили да су функције позване.
- Иницијализујте `i2c_driver` структуру и региструјте I²C руковаоца користећи исту. Уверите се да користите `compatible` подешавање које одговара истом које сте додали у Device Tree.



Линукс курс 2018/2019

Сада можете превести модул и поново покренути плочу да бисте је покренули са изменјеном DTB датотеком (уколико сте правили измене од претходног покретања).

ТЕСТИРАЊЕ РУКОВАОЦА

Сада можете учитати `/root/nunchuk/nunchuk.ko` датотеку. Проверите да ли је тада позвана функција `probe()`, и да ли је функција `remove()` позвана када уклоните модул.

ПРЕТРАГА /SYS

Уложите мало времена у претраживање `/sys`:

- Пронађите репрезентацију вашег рукаоца. То је и начин проналажења одговарајућих уређаја.
- Пронађите репрезентацију вашег уређаја која садржи његово име. Пронађи ћете и везу (link) до рукаоца

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan10 završen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Коришћење I²C магистрале

ЦИЉ

Оспособити функционалност I²C магистрале и искористити је за комуникацију са Nunchuk уређајем.

ИСХОД

Након ове вежбе ћете моћи да:

- Дефинишете подешавања pinmux-а
- Приступите регистрима I²C уређаја кроз магистралу.

ПОСТАВКА

Позиционирајте први терминал на путању `~/linux-kernel-labs/src/linux` ради превођења језгра и DTB (останите на `nunchuk branch-y`). Позиционирајте други терминал на путању `~/linux-kernel-labs/modules/nfsroot/nunchuk` ради превођења модула. Не заборавите да у оба терминала поставите променљиве `ARCH` и `CROSS_COMPILE` на одговарајуће вредности.

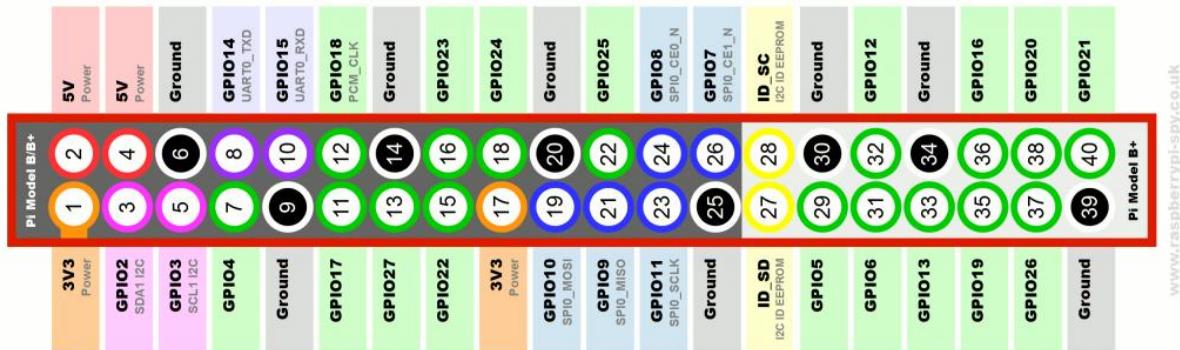
ПРОНАЛАЖЕЊЕ ИНФОРМАЦИЈА О КОНФИГУРАЦИЈИ МУЛТИПЛЕКСИРАЊА ПИНОВА ЗА I²C1

Како што смо видели у претходној вежби, добили смо nunchuk уређај пријављен на i2c1 магистралу.

Међутим, да бисмо приступили сигналима за податке и такт, морамо подесити и мултиплексирање пинова на SoC-у.

Линукс курс 2018/2019

Ако погледате слику испод, видећете да су пинови 3 и 5 (SDA1 и SCL1) конектора J8 заправо пинови GPIO2 и GPIO3, односно BCM2 и BCM3 (<http://pinout.xyz/>) на BCM2836 (или BCM2709) SoC-у. У документу BCM2835 ARM Peripherals у поглављу 6 General Purpose I/O (GPIO) (<https://github.com/raspberrypi/documentation/blob/master/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf>) ћете за пинове GPIO2 и GPIO3 видети да морају имати одабрану алтернативну функцију ALTO да би имали улогу пинова SDA1 и SCL1 периферије BSC1 периферије. То је доволно информација неопходних да урадите подешавање pinmux-а закоришћење BSC1 периферије (контролера), у нашем случају i2c1 чвора.



Слика 1 Распоред пинова на конектору 40-пинском конектору J8 на Raspberry Pi 2 плочи

У истом документу можете видети да GPIO пинови омогућавају укључивање интерних pull-up и pull-down отпорника, као и предвиђено подешавање за сваки у колони PULL (предвиђено је High односно pull-up за оба пина, мада се могу користити и без интерног pull-up отпорника јер за поменуте пинове већ постоје екстерни, на плочи).

Такође, у истом поглављу можете видети и да се GPIO периферија (која је заправо комбинација pinmux и interrupt контролера) налази на адреси 0x7E200000, као и информацију о сваком контролном регистру ове периферије (одстојање од базне адресе и улогу коју има са детаљном дефиницијом сваког од 32 бита).



Линукс курс 2018/2019

ДОДАВАЊЕ PINMUX ПОДЕШАВАЊА У DEVICE TREE

Када знамо позиције регистира, покушајмо да схватимо како се оне користе у постојећем коду. Отворите Device Tree за SoC-ове засноване на BSCM2708 (arch/arm/boot/dts/bcm2708_common.dtsi), који посредством bcm2709.dtsi датотеке укључује и већ поменута bcm2709-rpi-2-b.dts DTS датотека. У датотеци bcm2708_common.dtsi ћете осим дефиниције i2c1 чвора као потомка SoC чвора пронаћи и gpio чвор (pinmux контролер) са основним подешавањима, укључујући базну адресу ове периферије (0x7e200000).

У датотеци bcm2709-rpi-2-b.dts се налазе додатна подешавања ове периферије која су ту измештена јер одговарају само bcm2709 SoC-у, односно Raspberry Pi 2 плочи. За сваку од периферија којој је потребно дефинисати пинове које користи се на овом месту дефинише чвр потомак чвора gpio, па тако треба урадити и за i2c1 периферију (у оригиналној верзији овог документа је то и било дефинисано па можете искористити да одатле копирате или проверите оно што сте урадили).

Касније се референца на овај чвр користи у пољу pinctrl-0 одговарајуће периферије (i2c1 у нашем случају).

Могућа подешавања за pinmux контролер односно gpio чвр (а која користи сам gpio руковаљац) су:

Све опције које се могу користити можете пронаћи у документацији BCM2835 GPIO модула (нпр. <https://www.kernel.org/doc/Documentation/devicetree/bindings/pinctrl/brcm,bcm2835-gpio.txt>). За ову вежбу је у поменутом чвр потомку чвра gpio довољно подесити опције у наставку.

Обавезна подешавања чвра потомка:

- **brcm,pins:** Низ ћелија. Свака ћелија садржи ID pina. Валидни ID су целобројни делови GPIO ID; 0==GPIO0, 1==GPIO1, ... 53==GPIO53.

Опциона подешавања чвра потомка:



Линукс курс 2018/2019

- **brcm,function:** Цео број који представља функцију мултиплексирања пина/пинова:

0: GPIO in

1: GPIO out

2: alt5

3: alt4

4: alt0

5: alt1

6: alt2

7: alt3

- **brcm,pull:** Цео број који представља примену pull-down/up на pin/pinove:

0: none

1: down

2: up

На основу горенаведеног, чвр потомак чвора `gpio` који треба додати у датотеку `bcm2709-rpi-2-b-custom.dts` треба да изгледа овако:

```
&gpio {  
    i2c1_pins: i2c1 {  
        brcm,pins = <2 3>; /* GPIO2 и GPIO3 */  
        brcm,function = <4>; /* alt0 */  
    };  
};
```



Линукс курс 2018/2019

Такође, подешавања пинова сада треба увезати и у чвор `i2c1` који представља нашу периферију, односно дефинисати да `i2c1` користи управо те пинове:

```
pinctrl-names = "default";  
pinctrl-0 = <&i2c1_pins>;
```

Приликом додавања подешавања у DTS датотеку, не заборавите да подешавања која већ постоје у DTS1 датотекама не морате понављати и у DTS (нпр. основна подешавања за `gpio` чвор смо видели да постоје, додајемо само подешавања везана за пинове `i2c1` периферије тако што се референцирамо на постојећи чвор `gpio` као `&gpio`). Слично смо радили и за чвор `i2c1`, с обзиром да је и његова основна дефиниција у DTS1 датотеци. Дакле додајемо само нове декларације, односно подешавања која треба да замене стара, заједничка за фамилију процесора.

Поново преведите све DTB датотеке, прекопирајте и на крају поново покрените плочу.

ТЕСТИРАЊЕ I²C МАГИСТРАЛЕ

За тестирање магистрале ћемо користити `i2cdetect` команду да бисмо се уверили да све ради како је очекивано за `i2c1`:

```
# i2cdetect -l  
i2c-1      i2c      3f804000.i2c      I2C adapter
```

Уколико не видите поменути адаптер, вероватно модул `i2c_dev` није укључен (можете проверити командом `lsmod`). Модул можете додати командом

```
modprobe i2c_dev
```

Да не бисте сваки пут морали да додајете овај модул, једноставније је да га укључите у кернел:

```
make xconfig
```



Линукс курс 2018/2019

Претражите опцију I2C device interface (I2C_CHARDEV) и укључите је. Поново преведите кернел, прекопирајте zImage и поново покрените плочу. Након тога поновите претходни корак како бисте се уверили да је детектован i2c-1 адаптер.

Ако се и даље не види адаптер, могуће је да сте направили грешку у DTS датотеци.

```
# i2cdetect -F 1

Functionalities implemented by bus #1

I2C                      yes
SMBus quick command      yes
SMBus send byte          yes
SMBus receive byte       yes
SMBus write byte         yes
SMBus read byte          yes
SMBus write word         yes
SMBus read word          yes
SMBus process call       yes
SMBus block write        yes
SMBus block read          no
SMBus block process call  no
SMBus PEC                  yes
I2C block write           yes
I2C block read            yes
```



Линукс курс 2018/2019

Уколико приметите да SMBus Quick Commands нису доступне, а обзиром да их `i2cdetect` користи на I²C магистрале, можете помоћу `i2cdetect -r` користити уобичајени скуп I²C команди, и тиме детектовати све уређаје на магистрале.

Уверите се да све ради како је очекивано, покрените `i2cdetect 1`. Ово би требало да скенира све уређје на `i2c1` магистрале. Требало би да видите уређај на адреси `0x52`. То је ваш nunchuk.

ИНИЦИЈАЛИЗАЦИЈА УРЕЂАЈА

Следећи корак је читање стања nunchuk регистра, да бисмо нпр. проверили да ли је неки тастер притиснут или не.

Пре него што будемо у стању да читамо nunchuk регистре, прва ствар коју морамо урадити је да му пошаљемо иницијализационе команде. То је уједно и згодан начин да се уверимо да I²C комуникација исправно ради.

У функцији `probe()` (покреће се сваки пут када је одговарајуи уређај детектован):

- Користећи I2C raw API (на слајдовима са предавања), пошаљите уређају 2 бајта: `0xf0` и `0x55`¹. Уверите се да проверавате повратну вредност функција које користите. Оне би могле да открију евентуалне проблеме у комуникацији. У документацији на Интеренту потражите примере како исправно рукувати грешкама користећи исту функцију.
- Омогућите процесору да сачека 1 ms користећи функцију `udelay()`. По потреби, проверите која заглавља је потребно укључити за ову функцију.

¹ I²C поруке за комуникацију са nunchuk-ом су у sledećem формату: <i2c_address> <register> за чitanje, односно <i2c_address> <register> <value> за pisanje. Adresa `0x52` se šalje od strane okruženja, tako da je jedino što morate da uradite upis ostalih bajtova, adrese registra i, ukoliko je potrebno, vrednosti koju upisujete. Posotoje dva načina za uspostavu komunikacije. Prvi poznat način je sa enkripcijom upisom `0x00` u register `0x40` nunchuk-a. Na ovaj način morate dekriptovati svaki bajt koji pročitate iz nunchuk-a (nije komplikovan, ali jeste posao koji zhteva vreme). Na žalost, takva enkripcija ne radi на neoriginalnim nunchuk uređajima, па morate umesto toga koristiti komunikaciju bez enkripcije, коју postizete upisom `0x55` u register na adresi `0x00`. Ovo radi sa svim brendovima nunchuka (uključujući originalne Nintendo).



Линукс курс 2018/2019

- На исти начин, пошаљите сада још и байтове 0xfb и 0x0. Овим је завршена иницијализација nunchuk-a.

Поново преведите и учитајте модул. Уверите се да не постоје комуникационе греше, односно поруке које би биле исписане у случају грешке.

ЧИТАЊЕ NUNCHUK РЕГИСТARA

Nunchuk показује једно доста чудно понашање: делује као да се стање интерних регистара освежава само када се догоди читање стања регистрара.

Као последица тога, неопходно је да два пута прочитамо регистре којима желимо да проверимо стање.

- Да би код задржали једноставним и читљивим, направимо функцију `nunchuk_read_registers` која ће једном прочитати регистре. У овој функцији:
- Почните са паузом од 10 ms позивом функције `mdelay()`. То је трајање неопходне паузе између две узастопне I²C операције.
- Упишите 0x00 на магистралу. То ће омогућити читање стања регистрара.
- Додајте још једну паузу од 10 ms.
- Прочитајте 6 байтова са уређаја, и даље користећи I2C raw API. Проверите повратне вредности као и до сада.

ЧИТАЊЕ СТАЊА NUNCHUK ТАСТЕРА

Вратимо се на функцију `probe()`, где вашу нову функцију треба да позовете два пута.

Након другог позива, одредите стање Z и C тастера, које се може пронаћи у шестом байту који сте прочитали.



Линукс курс 2018/2019

Како што је објашњено у документу <http://www.robotshop.com/media/files/PDF/inex-zx-nunchuck-datasheet.pdf>:

бит 0 == 0 означава да је тастер Z притиснут.

бит 0 == 1 означава да тастер Z НИЈЕ притиснут.

бит 1 == 0 означава да је тастер C притиснут.

бит 1 == 1 означава да тастер C НИЈЕ притиснут.

Користећи се логичким операторима, напишите код који иницијализује `pressed` целобројну променљиву, чија вредност треба да буде 1 када је тастер Z притиснут, а 0 у супротном. Слично урадите и за променљиву `pressed` за тастер C².

Последње што треба урадити је провера стања нових променљивих на kraju функције `probe()` и испис поруке о томе који је тастер притиснут, у конзолу.

ТЕСТИРАЊЕ

Преведите ваш модул и учитајте га поново. Требало би да је детектовано да ни један тастер није притиснут. Уклоните модул.

Сада држите притиснут тастер Z и поново учитајте и уклоните модул:

```
insmod /root/nunchuk/nunchuk.ko; rmmod nunchuk
```

Сада би требало да видите поруке које потврђују да је руковаљац открио да је тастер Z био притиснут.

Урадите исто више пута са различитим стањем тастера.

У овој фази смо се само уверили да можемо да читамо стање регистара уређаја кроз I²C магистралу. Наравно, учитавање и укљањање модула сваки пут није згодан начин за приступ таквим подацима. На следећим вежбама ћемо додати руковаоцу одговарајућу улазну спрегу с обзиром да је у питању улазни уређај.

² Можете користити `BIT()` макро, који ће вам олакшати рад. Потражите документацију за овај макро на Internetu.



Линукс курс 2018/2019

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan11.1 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.



Линукс курс 2018/2019

Спрега са улазним подсистемом

ЦИЉ

Омогућити приступ I²C уређају из user space-а коришћењем улазног подсистема.

ИСХОД

Након ове вежбе ћете моћи да:

- Омогућите приступ догађајима user space-у кроз улазни подсистем (спрегу) коришћењем polling API језгра оперативног система Линукс за улазне уређаје (kernel space перспектива).
- Рукујете грешкама приликом регистрације или алокације на коректан начин.
- Користите улазну спрегу са више разумевања и више детаља (user space перспектива).

ПОСТАВКА

Позиционирајте први терминал на путању ~/linux-kernel-labs/src/linux ради превођења језгра и DTB (останите на истом branch-y). Позиционирајте други терминал на путању ~/linux-kernel-labs/modules/nfsroot/nunchuk ради превођења модула. Не заборавите да у оба терминала поставите променљиве ARCH и CROSS_COMPILE на одговарајуће вредности.

ДОДАВАЊЕ ПОДРШКЕ ЗА POLLING УЛАЗНИ УРЕЂАЈ У КЕРНЕЛ

Nunchuk нема прекиде (interrupt) којима би обавестио I²C мастер да је његово стање промењено. Стога је једини начин за приступ подацима уређаја и



Линукс курс 2018/2019

детектовање промена прозивање (polling) његових регистара коришћење улазног polling API описаног на предавањима.

Поново преведите кернел са статичком подршком за polled input device (`CONFIG_INPUT_POLLDEV=y`) и за event interface (`CONFIG_INPUT_EVDEV`). У подразумеваној конфигурацији, они су доступни као модули, што је мање погодно. Прекопирајте `Image` датотеку и покрените поново плочу.

РЕГИСТРОВАЊЕ УЛАЗНЕ СПРЕГЕ

Прво што треба урадити је додавање улазног уређаја у систем. То је могуће урадити у следећим корацима:

- Декларисати показивач на `input_polled_dev` структуру у функцији `probe`. Можете га назвати `polled_input`. Није погодно користити глобалну променљиву јер ваш руковалац треба да има подршку за више уређаја прикључених истовремено.
- Алоцирајте поменуту структуру у истој функцији, користећи функцију `input_allocate_polled_device()`.
- Такође, дефинишиште показивач на `input_dev` структуру. Можете га назвати `input`. Алокација није потребна јер је он већ део `input_polled_dev` структуре и истовремено је алоциран. Користићемо га само као пречицу да бисмо код одржали што једноставнијим.
- У функцији `probe()` додајте улазни уређај у систем позивом `input_register_polled_device()`.

У овој фази се најпре уверите да се ваш модул успешно преводи (по потреби додајте недостајућа заглавља).



Линукс курс 2018/2019

РУКОВАЊЕ ГРЕШКАМА У PROBE ФУНКЦИЈИ

У коду који сте написали, проверите да ли на одговарајући начин реагујете у случају могућих грешака:

- Пре свега, увек на одговарајући начин проверите повратне вредности функција и евидентирајте узроке грешака.
- Ако је позив функције `input_register_polled_device()` неуспешан, морате ослободити `input_polled_dev` структуру пре него што вратите код грешке, односно пре него што се заврши извршавање функције. Ако то не урадите створићете цурење меморије (*memory leaks*) у кернелу. Уопштено говорећи, неуспешно отпуштање онога што је претходно било алоцирано или регистровано може довести до проблема односно може спречити поновно учитавање модула.

Да бисте ово коректно имплементирали без сувишног копирања кода или рујног и непрегледног кода, ово је место где се препоручује употреба `goto` наредби.

ИМПЛЕМЕНТАЦИЈА ФУНКЦИЈЕ REMOVE()

У овој функцији је потребно отпустити све ресурсе које је заузела или регистровала функција `probe()`.

Међутим, ово није тривијално као што је била имплементација функције `probe()`. Како наш руковалац треба да подржава више уређаја истовремено, одлучили смо се да не користимо гобалне променљиве, а као последица тога сада не можемо користити такве променљиве да бисмо ослободили одговарајуће ресурсе.

Ово доводи до веома важног аспекта модела уређаја: потребе да се показивачи одрже између физичких уређаја (уређаја којима рукује физичка магистрала, I²C у нашем случају) и логичких уређаја (уређаја којима рукују подсистеми, као што је улазни подсистем у нашем случају).



Линукс курс 2018/2019

На овај начин, када је позвана функција `remove()` (типично када магистрала детектује да је неки од уређаја уклоњен) можемо открити који је логички уређај потребно одјавити (`unregister`). Обрнуто, када имамо догађај на логичкој страни (као што је отварање или затварање улазног уређаја по први пут), можемо открити којем I^2C slave уређају тај догађај припада, како бисмо урадили специфичне кораке са физичким уређајем (hardware).

Ово је типично имплементирано кроз стварање структуре приватних података за управљање уређајем и имплементацију показивача између физичког и логичког света.

Додајте следеће дефиниције у ваш код:

```
struct nunchuk_dev {  
    struct input_polled_dev *polled_input;  
    struct i2c_client *i2c_client;  
};
```

Затим у функцији `probe()`, декларишите инстанцу ове структуре:

```
struct nunchuk_dev *nunchuk;
```

Потом алоцирајте једну такву структуру за сваки нови уређај:

```
nunchuk      =      devm_kzalloc(&client->dev,      sizeof(struct  
nunchuk_dev), GFP_KERNEL);  
  
if (!nunchuk) {  
    dev_err(&client->dev, "Failed to allocate memory\n");  
    return -ENOMEM;  
}
```

Приметите да до сада нисмо видели функције за алоцирање меморије у кернелу.

Све што за сада треба да знате је да се свака алокација или регистрација коју направи `devm_` функција припаја структури уређаја (device structure). Када се



Линукс курс 2018/2019

уређај или цео модул уклоне, све такве алокације или регистрације се аутоматски пониште. Ово у значајној мери поједностављује код руковаоца.

Још увек нисмо објаснили ни функције `dev_*` за евидентирање порука (оне се у основи користе да би јасно истакле за који уређај је евидентирана порука везана).

Сада имплементирајте показиваче:

```
nunchuk->i2c_client = client;  
nunchuk->polled_input = polled_input;  
polled_input->private = nunchuk;  
i2c_set_clientdata(client, nunchuk);  
input = polled_input->input;  
input->dev.parent = &client->dev;
```

Водите рачуна да овај код треба да стоји пре регистраовања улазног уређаја. Не би требало омогућити рад уређаја са непотпуним информацијама, односно док све информације нису уписане (у супротном би могло доћи до трке).

Када је све ово урађено, коначно имамо све што је неопходно да би се имплементирала функција `remove()`. Потражите примере на слајдовима о I²C и улазним уређајима или директно у Линукс кернел коду!

Преведите поново модул, учитајте га, па уклоните неколико пута, како бисте се уверили да је све успешно пријављено односно одјављено.

ДОДАВАЊЕ ИСПРАВНИХ ИНФОРМАЦИЈА О РЕГИСТРАЦИЈИ УЛАЗНОГ УРЕЂАЈА

Пре саме регистрације улазне структуре, морамо јој додати још информација. То је разлог због којег у овом тренутку још увек добијате упозорења приликом учитавања модула, односно пријаве новог уређаја:

```
input: Unspecified device as /devices/virtual/input/input0
```



Линукс курс 2018/2019

Додајте линије кода које следе (такође пре регистрације, наравно):

```
input->name = "Wii Nunchuk";  
  
input->id.bustype = BUS_I2C;  
  
set_bit(EV_KEY, input->evbit);  
  
set_bit(BTN_C, input->keybit);  
  
set_bit(BTN_Z, input->keybit);
```

Поново преведите модул и покушајте поново да га учитате. Сада би требало да видите међу кернел порукама да је „Unspecified device type“ замењено са „Wii Nunchuck“ и да је физичка путања до уређаја такође пријављена.

ИМПЛЕМЕНТАЦИЈА POLLING РУТИНЕ

Остало је још да имплементирамо функцију која ће прозивати nunchuk регистре у предвиђеном временском интервалу.

Направите функцију `nunchuck_poll()` са одговарајућим прототипом (пронађите га по угледу на дефиницију `input_polled_dev` структуре).

Најпре додајте линије којима преузимате физички I²C уређај из `input_polled_dev` структуре. Ту ће вам бити потребна приватна `nunchuk` структура.

Сада када имате показивач (`handle`) ка физичком I²C уређају, код за читање `nunchuk` регистара можете пребацити у ову функцију. Можете уклонити двоструко читање стања уређаја јер ће polling функција свакако правити периодична читања¹.

На крају polling рутине, последње што треба урадити је постављање догађаја и обавештење језgra улазног подсистема. Под претпоставком да је `polled_input` назив `input_polled_dev` параметра ваше polling рутине:

¹Приликом пребацивања кода, мораћете да рукујете комуникационим грешкама на мало другачији начин, с обзиром да функција `nunchuk_poll()` има `void` тип. Када је функција за читање регистара неуспешно завршена, можете кориситити `return;` instead of `return value;`



Линукс курс 2018/2019

```
input_event(polled_input->input,  
EV_KEY, BTN_Z, zpressed);  
  
input_event(polled_input->input,  
EV_KEY, BTN_C, cpressed);  
  
input_sync(polled_input->input);
```

Вратите се на функцију `probe()`. Последње што требада урадите је да декларишете нову polling функцију (погледајте слајдове са предавања уколико сте заборавили детаље) и наведете polling интервал од 50 ms.

Проверите да ли се код преводи и учитава.

ТЕСТИРАЊЕ УЛАЗНЕ СПРЕГЕ

Тестирање улазног уређаја је једноставно када се користи `evtest` апликација која је укључена и у root filesystem.

Само покрените:

```
evtest
```

Апликација ће вам приказати све доступне улазне уређаје и дозволити вам да одаберете један по свом избору (обавезно одаберите један, иако је подразумевано 0 немојте само притиснути [Ентер], већ претходно упишите свој избор).

Можете одмах да тестирате и на следећи начин

```
evtest /dev/input/event0
```

Притисните различите тастере (комбинације) и уочите како су одговарајући дугачки пријављени кроз евтест апликацију.



Линукс курс 2018/2019

ДОДАТНИ ЗАДАТAK

Уколико вежбу завршите пре других, можете додати подршку и за координате nunchuk joystick-a.

Друга ствар коју можете урадити је додавање подршке за координате nunchuk акцелерометра.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A  
git commit -as -m "dan11.2 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.