|  |  |  |
| --- | --- | --- |
|  | **УНИВЕРЗИТЕТ У НОВОМ САДУ**  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** |  |

|  |
| --- |
| **УНИВЕРЗИТЕТ У НОВОМ САДУ**  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  **НОВИ САД**  **Департман за рачунарство и аутоматику**  **Одсек за рачунарску технику и рачунарске комуникације**  **ИСПИТНИ РАД**  **Кандидати: Стеван Стевић**  **Број индекса: РА74/2014**  **Предмет: Системска програмска подршка у реалном времену II**  **Тема рада: Raspberry Pi 2 Linux руковалац за сензор вибрације SW-420**  **Ментор рада: Мирослав Поповић**  **Нови Сад, јануар, 2017.** |
|  |

Садржај

[1. Увод 1](#_Toc473164385)

[1.1 Задатак 1](#_Toc473164386)

[1.2 Анализа проблема 1](#_Toc473164390)

[2. Концепт решења 2](#_Toc473164391)

[3. Опис решења 3](#_Toc473164392)

[3.1 Функција за иницијализацију модула 3](#_Toc473164393)

[3.2 Функција за завршетак рада са модулом 3](#_Toc473164394)

[3.3 Фукција за отварање 4](#_Toc473164395)

[3.4 Фукција за затварање/отпуштање 4](#_Toc473164396)

[3.5 Фукција за упис/слање 4](#_Toc473164397)

[3.6 Функција за читање/пријем 5](#_Toc473164398)

[3.7 Прекидна рутина 5](#_Toc473164399)

[3.8 Callback функције временске контроле 6](#_Toc473164400)

[4. Закључак 7](#_Toc473164401)

[4.1 Концепт тестирања 7](#_Toc473164402)

[4.2 Резултати тестирања 7](#_Toc473164403)

# 1. Увод

## 1.1 Задатак

## Потребно је реализовати Linux руковалац за Raspberry Pi 2 уређај који треба да омогући комуникацију корисничког програма и сензора који је повезан на GPIO пролаз доступан на уређају. Дигитални излаз са сензора означава присуство/одсуство вибрација.

## У сврху провере функционалности развијеног руковаоца потребно је развити корисничку апликацију која посредством имплементираног модула може да добави информације о вибрацијама које изворно потичу од сензора, то јесте да добави сигнал четвртки који руковаоц генерише путем одабирања, а затим одради дискретну Фуријеову трансформацију и израчуна фреквенцију вибрације. Такође омогућити да апликација добави само периоду фреквенције.

## Апликација треба да омогући рад са произвољаним бројем сензора вибрације (1-4) и да паралелно обрађује и иписује информације пристигле са сензора. Свако језгро треба да преко једног уређаја тј. инстанце руковаоца преузима подаке са сензора, и прерачуна фреквенцију или испише периоду.

## 1.2 Анализа проблема

Један од главних проблема при имплементацији овог руковаца јесте омогућити руковање са више (конкретно 4) сензора у исто време. Потребан је по један чвор уређаја за сваки сензор, преко којег тест апликација може комуницирати са сензорима. Комуникација се обавља преко функције read(), јер је за апликацију битно само да прима податке о вибрацији у облику периоде или сигнала.

Следећи проблем јесте само очитавање сензора које се мора обављати преко рутине за обраду прекида (енг. interrupt service routine). Потребно је имплементирати прекидну рутину која обрађује сваки прекид који сензор изазове, код рачунања периоде. Као и *callback* функцију за времеснки бројач код одабирања сигнала и подесити исправну фреквенцију. Проблем код оба начина рада јесте обезбедити поменутим функцијама да знају са којим сензором раде и којем чвору прослеђују податке. На крају потребно је путем read() функције проследити податке апликацији, односно кориснику.

У тестној апликацији потребно је омогућити кориснику да изабере начин на који ће радити read() функција. Односно да ли ће при позиву враћати периоду или сигнал. Ако корисник изабере да као повратну вредност добије сигнал, такође може и да зада фреквенцију одабирања и израчуна фреквенцију путем ДФТ[[1]](#footnote-1).

Што се тиче хардверске стране, потребно повезати сензоре на одговарајуће ГПИО пинове Raspberry Pi 2 платформе. Конкретно потребно је повезати Vcc (5V - 3,3V), GND (eng. Ground) и DO (eng. Digital Output). Vcc представаља напајање за сензор, GND је маса, а DO сам податак да ли је сензор детектовао вибрацију (може бити 0 или 1).

# 2. Концепт решења

Да би се омогућио исправан рад руковаоца, потребно је заузети слободан главни број, као и нагласити ОС-у колико сензора је могуће повезати приликом регистровања руковаоца преко devices\_to\_create променљиве. Главни број је једноставно решење помоћу којег ОС зна који који руковалац обрађује које периферне догађаје, док споредни бројеви представљају сам уређај који користи тај руковалац. То је одрађено у иницијализационој функцији, кориштењем већ постојећих функција. Руковалац такође сам у овој фази прави и чворове, koji se налазе у /dev/ именику и зову се **vibrato0/3** преко којих корисник комуницира са руковаоцем, односно сензором.

Ако се све успешно заврши, руковалац почиње да на сваки прекид који прими од сензора, а који се јавља када сензор региструје вибрацију и на DO пин пошаље 1, рачуна периоду на основу протеклог времена између два прекида. Прецизно време се добија употребом *hr\_timer* – а.

Када корисник, руковалац пребаци да ради у режиму за одабирање сигнала, тада се тек приликом позива функције read() покреће временски бројач који је подешен да одбројава периоду која се израчунава на основу задате фреквенције. Унапред подешена фреквенција износи 8 kHz. По истеку бројача позива се *callback* функција у којој се очитава тренутна вредност пина и тиме добија изглед сигнала. Сигнал се састоји од поворке четвртки јер сензор има дигитални излаз, односно шаље 0 или 1.

У зависноти од начина рада руковалац у функцији за читање копира периоду или сигнал из меморијског простора језгра (кернела) у кориснички, одакле корисник може даље радити са подацима. Руковаоцу није потребна функција за писање јер не треба ништа да прослеђује руковаоцу.

Бирање режима рада се обавља преко стандарне функције за контролу руковаоцима *ioctl().* Њој се прослеђује 0 и жељена фреквенција одабирања ако корисник жели сигнал као повратну вредност, а 1 ако је жељени податак периода.

При отварању чвора помоћу функције open() иницијализује се „уређај“, који представља везу између корисника и сензора. Због могућности истовремног повезивања више сензора, ти уређаји су смештени у низ који је типа *struct device\_st,* oодносно структура која у себи садржи сва поља карактеристична за један „уређај“ и његово исправно функционисање.

# 3. Опис решења

За покретање и коришћење модула потребно га је прво регистровати (insert-овати) у језгро. То се извршава командом "insmod vibrations\_module.ko" приликом чега се позива функција у оквиру модула за његову иницијализацију, или са параметрима "insmod vibrations\_module.ko devices\_to\_create=3 di\_GPIO\_pins=7, 9, 12"

## 3.1 Функција за иницијализацију модула

int vibrations\_module\_init(void);

Као што је већ напоменуто ова функција се позива приликом регистровања модула руковаоца у језгро и служи да се сви параметри руковаоца правилно поставе пре самог коришћења истог. Иницијализација параметара обухвата исправну обраду примљених аргумената (који могу бити GPIO пинови на које се каче DO пинови са сензора и број сензора), затим заузимање главног броја и заузимање региона, иницијализација временске контроле, постављање пинова Raspberry Pi 2 уређаја на који су накачени сензори, да могу направити прекид као и да се понашају као улазни пинови и само креирање чворова. Ако се све успешно заврши повратни код је 0, а у супротном број грешке и њен опис, који се може видети излиставањем кернел порука.

## 3.2 Функција за завршетак рада са модулом

void serial\_driver\_exit(void);

Функција за завршетак рада са модулом се позива приликом брисања/извлачења руковаоца из језгра командом "rmmod vibrations\_module".

У њој се одјављује модул из кернела, а затим и уништавају претходно креирани чворови. Осим тога још се ослобађа и одговарајући канал за прекиде, као и отказивање временских контрола, како би руковалац успешно био уклоњен. Такође врши се и уништавање уређаја, који служе као спона између апликације и сензора.

## 3.3 Фукција за отварање

int vibration\_driver\_open(struct inode\* inode, struct file\* filp);

* + inode – структура која садржи поља везана за уређај
  + filp - показивач на датотеку

Функција за отварање уређаја служи како би иницијализовала сва поља структуре и заузела сву потребну меморију. То укључује једноставно постављање вредности одређених поља на нулу, као и иницијилаизацију временске контроле и *mutex* – a.

## 3.4 Фукција за затварање/отпуштање

int vibration\_driver\_release(struct inode\* inode, struct file\* filp);

Параметри су исти као и код претходне функције. У овој функцији потребно је само прекинути временску контролу која се иницијализује у функцији за отварање. Ово служи више као мера заштите јер се котнтрола поништава у самој функцији за читање где се дешава одабирање.

## 3.5 Фукција за упис/слање

static ssize\_t vibrations\_module\_write(struct file\* filp, const char\* buf, size\_t len, loff\_t\* f\_pos);

Функција за упис/слање би се позивала приликом уписивања у на пример специјалну датотеку /dev/vibrat0 и приликом тога би долазило до слања података. Повратна вредност ове функције указује на то колико је знакова послато, а њени аргументи су :

* + filp - показивач на датотеку
  + buf - показивач на бафер из кога копирамо податке из корисничког простора (*user space*)у кернел
  + len - број бајтова који уписујемо/шаљемо
  + f\_pos – почетни померај у датотеци

Ова функција није имплементирана у овом руковаоцу, јер није било потребе за њеним кориштењем. Кориснику је довољно само да прима податке од руковаоца.

## 3.6 Функција за читање/пријем

static ssize\_t vibrations\_module\_read(struct file\* filp, char\* buf, size\_t len, loff\_t\* f\_pos);

Функција за читање/пријем се позива приликом читања горе наведене специјалне датотеке и приликом тога долази до пријема података. Повратна вредност ове функције указује на то колико је података примљено. Аргументи функције су слични са тим да сада buf представља показивач на бафер у који се уписује ради преноса података из кернела у кориснички простор.

Такође у овој функцији се проверава да ли је режим рада подешен за враћање периоде или сигнала. На основу те провере се враћа периода или отпочиње одабирање сигнала по задатој фреквенцији, одређен број одбирака (BUFF\_LEN) kоји је подешен на вредност од 4096 одбирака како би FFT[[2]](#footnote-2) био степена двојке и како би било довољно одбирака да се што прецизније одреди периода.

## 3.7 Прекидна рутина

static irqreturn\_t h\_irq\_gpio(int irq, void \*data);

Прекидна рутина рачуна тренутно периоду сигнала на основу времена које је прошло између 2 прекида. Она одмах примењује и ***IIR - moving average*** филтар како би вратила што бољу и прецизнију вредност.

Код примене IIR filtra se javlja problem jer његова формула гласи:

Где α има вредности између 0 и 1, што значи да се добијају реалне вредности, које у кернел програмирању нису подржане. Овај проблем је решен тако што су те вредности скалиране на вредности *short int -*  a јер је променљива која представља период тог типа, а то је распон од 0 – 32767. Резултат се смешта у 32-битну променљиву, а затим врши померање за 16 места у десно како би се добила права вредност.

## 3.8 Callback функције временске контроле

enum hrtimer\_restart period\_timer\_callback(struct hrtimer \*param);

Ова функција се позива по истеку времеснке контроле која броји једну милискеунду. У њој се увећава глобална променљива која представља време и на основу ње се рачуна периода.

enum hrtimer\_restart sampling\_timer\_callback(struct hrtimer \*param);

Функција коју користи временска контрола која одбројава време за које треба очитати стање пина на основу задате периоде. Највећи проблем је сазнати које уређај је затражио сигнал, како би се у његов бафер уписале очитане вредности. То је решено помоћу функције *container\_of()* која враћа показивач на потребни уређај, а то ради тако што лоцира у којој се структури налази прослеђени *param*.

Након тога потребно је једноставно очитати вредност пина помоћу функције *GetGpioPinValue()* сместити ту вредност у бафер и увећати променљиву *sample* која представља тренутни одбирак и понаша се као индекс бафера.

# 4. Закључак

## 4.1 Концепт тестирања

Тестирање се изводи тако што се сензор прислони на неки извор вибрације, руковалац постави у жељени режим и позове функција читања која ће вратити коресподентне податке. Потребно је учврстити сензор и поставити га на место где је извор вибрација најјачи како би резутати били што прецизнији. Некада је потребно више пута позвати функцију читања како би се добили тачни резултати, због неипсуњених горе поменутих услова.

## 4.2 Резултати тестирања

Резултати показују да се правилном применом FFT-a над добијеним сигналом може израчунати фреквенција вибрације. За пример је кориштена вибрација мобилног телефона. За различите моделе телефона фреквенција је била другачија, што је потврђивало да све ради исправно. Фреквенције су се кретале у распону од 200 – 300 Hz што и јесте реална ситуација.

Тачност периоде је теже тестирати јер је сензор веома осетљив, а пошто га и сама вибрација телефона додатно помера, то некада даје нетачне резулатате. Када би се сензор учврстио за константан извор (по могућности познате фреквенције) вибрације резултати би били у потпуности валидни.

1. ДФТ – Дискретна Фуријеова трансформација је Фуријеова трансформација дискретног и коначног (или периодичног) сигнала, која из временског спектра прелази у фреквенцијски. [↑](#footnote-ref-1)
2. FFT - Брза Фуријеова трансформација је алгоритам за „брзо“ израчунавање вредности дискретне Фуријеове трансформације [↑](#footnote-ref-2)