Зборник радова Факултета Техничких Наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/22BE29Novakovic

РЕАЛИЗАЦИЈА РЕПРОДУКОВАЊА СЛИКЕ НА *LED* ДИСПЛЕЈУ СА ПАСИВНОМ МАТРИЦОМ

REALIZATION OF PICTURE REPRODUCTION ON *LED* DISPLAY WITH PASSIVE MATRIX

Стефан Новаковић, *Факултет Техничких Наука, Нови Сад*

Област – ПРИМЕЊЕНА ЕЛЕКТРОНИКА

Кратак садржај – *У оквиру овог рада описан је рад LED дисплеја са пасивном матрицом и описана је софтверска реализација репродуковања слике на истом.*

Кључне речи: *LED дисплеј, Python, Raspberry Pi, C библиотека*

Abstract – *This paper describes working principle of LED display with passive matrix and software implementation of image reproduction on it.*

Keywords: *LED display, Python, Raspberry Pi, C library*

1. УВОД

Тема рада је реализација репродуковања слике на *LED* дисплеју са пасивном матрицом. Како смо свакодневно окружени различитим дисплејима, циљ овог рада је упознавање технолигије приказа слике. Иако коришћени дисплеј представља помало застарелу технологију за приказ садржаја, јер се не користи у рачунарима, мобилним телефонима, телевизорима и другим савременим уређајима, већ само као рекламни панели, за сценску кореографију или у возилима јавног превоза, ови дисплеји представљају добру основу за разумевање рада дисплеја. Постоје разне поделе дисплеја на основу њиховог начина адресирања, материјала које користе, начина рада итд. Оно што је заједничко сваком дисплеју је да се сваки пиксел састоји од три извора светлости, црвеног, зеленог и плавог извора светлости. Комбинацијом интезитета ова три извора светлости, може се добити широк спектар боја.

Осим мотивације да се више упозна начин приказивања слике на дисплеју, један од главних разлога зашто је изабрана ова тема је и мотивација за упознавањем са програмским језиком *Python*. *Python* представља моћан програмски језик, који своју примену налази у више различитих области рачунарства. Он може да се користи у роботици, аутоматизацији, *Web* апликацијама, анализи података, вештачкој интелигенцији и још многим другим областима.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Јован Бајић, ванр. проф.

2. *LED MATRIX* ДИСПЛЕЈ

У зависности од матрице приказа, чији су елементи светлосно-емитујући пиксели, дисплеје можемо поделити на активне и пасивне. Ова подела се односи на начин адресирања. Код дисплеја са пасивном матрицом сваки пиксел реагује (емитује светлост) само кад је адресиран, а у осталим интервалима је неактиван (мултиплексни режим рада). Дисплеји са пасивном матрицом представљају старију технологију и добри су само за статичан приказ (слика, текст и сл.). Главни недостаци ових дисплеја су споро време одзива, прелушавање између линија, мали максимални контраст и израда матрице може бити јако комплексна.

2.1. *LED* дисплеј са пасивном матрицом

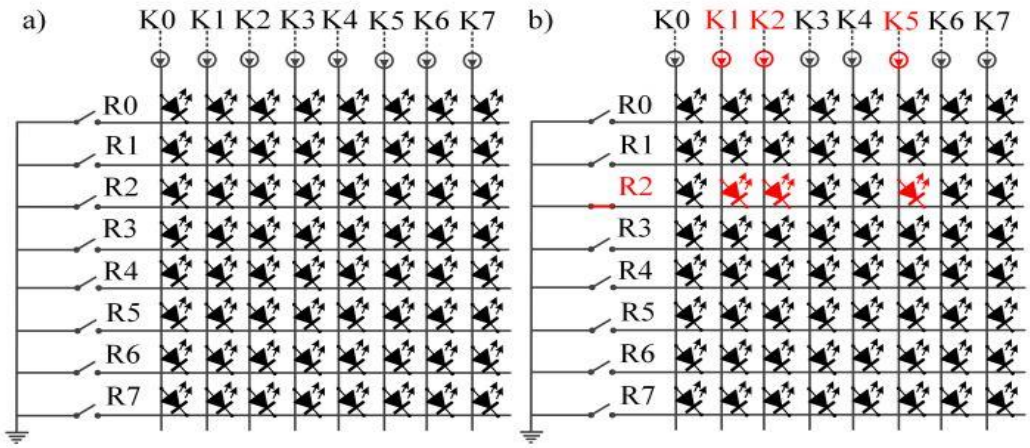
Редови диплсеја се реализују преко прекидача, који у пракси представљају транзисторе спојене на масу, а колоне на струјне изворе који служе за напајање *LED* диода. Испис садржаја на дисплеју врши се ред по ред. Спуштањем једног од прекидача затвара се струјно коло и селектује одговарајући ред. За *LED* диоде које треба да емитују светлост укључују се струјни извори у одговарајућим колонама.

LED диоде се држе у укљученом стању одређено T_{ON} време. Након тога, дати ред се искључује. Описани поступак се понавља за сваки следећи ред, почев од првог до последњег. Активирање струјних извора у колонама одређује који пиксели ће емитовати светлост у селектованом реду. Оваквој реализацији је неопходно стално освежавање садржаја дисплеја, како би се садржај константно приказивао.

При том, укупно време освежавања мора бити кратко како људско око не би регистровало треперење *LED* диода. Ово подразумева да дисплеј већих димензија захтева веће време освежавања. Диода која треба да емитује светлост биће укључена време T_{ON}, а искључена преостало време T_{OFF} до поновног укључења:

$$T_{OFF} = (\text{број редова} - 1) * T_{ON} \qquad (1)$$

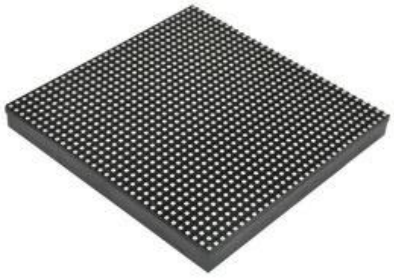
Ако је време освежења дисплеја (T_{ON}+T_{OFF}) довољно мало људско око неће бити у стању да региструје тренутке укључења и искључења *LED* диоде, већ ће регистровати средњи интезитет светлости одређен временим T_{ON} и T_{OFF}. Што је време T_{OFF} дуже у односу на T_{ON} средњи интезитет који региструје око биће мањи.



Слика 1. а) *8x8 LED дисплеј са пасивном матрицом б) 8x8 LED дисплеј са селектованим редом 2 и укљученим колонама 1, 2 и 5*

2.2. *LED* панел

За реализацију овог пројекта користило се 6 *LED* панела спојених редно. Сваки *LED* панел садржи 1024 пиксела распоређена у матрици од 32 реда и 32 колоне. Сваки пиксел се састоји од засебних црвених, зелених и плавих *LED* диода унутар истог кућишта.



Слика 2. *32x32 RGB LED панел*

LED панел је подељен вертикално на горњи (32 колоне и 16 редова) и доњи потпанел (32 колоне и 16 редова).

Горњи и доњи потпанел су такође вертикално подељени на по два сегмента, од по 32 колоне и 8 редова. Сегменти у оквиру истог панела су повезани редно. Колонама панела управља се помоћу једног сета управљачких сигнала, а редовима помоћу другог [1].

3. РЕАЛИЗАЦИЈА РЕПРОДУКОВАЊА СЛИКЕ

За хардверску реализацију овог пројекта коришћен је *Raspberry Pi* 4 и *LED* дисплеј који се састоји од шест панела 32x32 *HUB*-75 редно везаних.

Raspberry Pi 4 представља мини рачунар опште намене са могућношћу прикључивања нестандардне опреме. На *Raspberry Pi* 4 се налази оперативни систем *Raspbian OS* који је заснован на *Debian LINUX* оперативном систему и оптимизован је за рад са *Raspberry Pi* уређајима [2]. *LED* дисплеј се напаја посебним напајањем од 5V.

Сваки панел се састоји од два порта, улазног и излазног који служи за везивање више панела.

Сваки порт се састоји од улазних пинова *A, B, C, R1, G1, B1, R2, G2, B2, OE, LAT и CLK.*

3.1. Реализације апликације у *Python* програмском језику

Осим што *Raspberry Pi* промовише *Python* као главни програмски језик, један од главних разлога зашто је апликација првобитно реализована у том програмском језику је жеља да се научи *Python*. Разлог да се научи *Python* лежи у његовом широком спектру могућности, налазећи примену у роботици, аутоматизацији, развоју *Web* апликација, анализи података, вештачкој интелигенцији итд.

За имплементацију апликације најбитније функције су функције за контролу *GPIO* излаза:

- **selectRow(row)** - функција којом се врши селекција реда *LED* панела (постављање стања *A*, *B* и *C* улазних прикључака панела). Улазни параметар је редни број реда (0-7).
- **latch()** - функција којом се подаци пребацују у паралелни излазни регистар променом стања улазног прикључка *LAT* панела са "1" на "0".
- **clock()** - функција којом се генерише једна периода такт сигнала, позивањем ове функције врши се померање података у померачком регистру променом стања улазног прикључка *CLK* са "1" на "0".
- **setColor(color, bit)** - функција којом се поставља стање улаза *R1*, *G1*, *B1*, *R2*, *G2* и *B2 LED* панела. Улазни параметри су редни број бита променљиве *color* који се поставља на претходно наведене улазе.

Сигнал *OE* (енг. *Output Enable*), служи за укључивање и искључивање активних излазних регистара, који директно адресирају појединачне колоне. Осим функција за контролу функције *GPIO* излаза, битно је поменути и остале функције за исписивање садржаја, а то су: функције за цртање (*displayDrawRectangle*(), *displayDrawCircle*() итд.), функције за писање текста (*displayPutChar*(), *displayPrint*() итд.) и функција за приказивање слике (*displayLoadImage*()).

Главну функцију у реализацији представља функција *refreshDisplay*() која служи, као што јој и само име каже, за освежавање садржаја који се приказује на дисплеју. Идеја за апликације које приказују различит садржај на дисплеју је да се имплементација свих

потребних функција налази у једном модулу, док се свака апликација која приказује одређени садржај састоји од task-а који се позива само приликом иницијализације и цикличног task-а који позива само функцију *refreshDisplay*(). Функције које се налазе у иницијалном task-у уписују у низ *FRAME_BUFF* у зависности од тога шта треба да прикажу на дисплеју, а функција *refreshDisplay*(), као што је већ поменуто, приказује тај садржај на дисплеју. За управљање *GPIO* пиновима користила се *RPi.GPIO* библиотека. Све пинови, који су спојени са пиновима *LED* дисплеја, потребно је подесити као излазне пинове (*A, B, C, R1, G1, B1, R2, G2, B2, OE, LAT и CLK*).

Као што је већ и поменуто, да би се садржај стално приказивао на дисплеју, потребно је периодично понављати поступак исписивања садржаја, а да при том, укупно време освежавања буде кратко како људско око не би регистровало треперење. Што је већи дисплеј, то постаје изазовније. Приликом покретања апликације примећени су основни недостаци пасивних дисплеја: треперење и појава *ghosting* ефекта.

Слика 3. *Испис садржаја на дисплеју помоћу Python апликације*

Постоји више разлога појаве ових недостатака, један од главних је то што је *Python* програмски језик јако спор за извршавање. Други разлог лежи у томе што се на *Raspberry Pi* налази оперативни систем који одузима доста процесорске моћи. Време потребно да се изврши функција *refreshDisplay*() је реда ~85ms.

```
0.08538103103637695
0.0854494571685791
0.08484554290771484
0.08485221862792969
0.08454108238220215
0.08472323417663574
0.08444547653198242
0.08467674255371094
0.08456110954284668
0.08563470840454102
```

Слика 4. *10 узорака мерења времена извршавања функције refreshDisplay() са реализацијом у Python-у*

Постоји више начина за превазилажење ових проблема: коришћење *low-level* програмских језика (нпр. *C*), употреба *overclocking*-а (повећање фреквенције процесора), ослобађање *RAM* меморије (брисање непотребних апликација), додељивање већег приоритета апликацији (опсег приоритета је од -20 до 19, где је -20 највећи приоритет, док је по стандарду нашој апликацији додељен приоритет 0), употреба квалитетног извора напајања, употреба система хлађења, активација *ZRAM*-а итд.

3.2. Зашто је *Python* толико спор?

Python је динамички превођен језик. У програмским језицима као што су *C, C++* или *Java* све променљиве су статичке, то значи да програмер сам одређује тип за своју променљиву: *int my_var* = 1; У *Python*-у можемо једноставно да напишемо *my_var* = 1. Иако је ово веома згодно за програмера, има доста мана.

Компајлирање значи превођење једног програмског језика у други, обично у нижи ниво од превођеног језика. Када се компајлира програмски језик *C*, он се преводи у машински језик, који представља инструкције за сам процесор. *Python* се компајлира у *bytecode* јер је он динамички превођен језик. Ми не одређујемо тип за нашу променљиву, ми морамо да чекамо вредност променљиве како би одредили њен тип и тек онда превели у машински језик. Ово је оно што преводилац ради. У статички превођеним језицима, компајлирање и интерпретација се дешава пре извршавања програма. Закључак: програм је успорен компајлирањем и интерпретацијом која се дешава у току извршавања самог програма [3].

4. РЕАЛИЗАЦИЈА *C* БИБЛИОТЕКЕ

Као што смо већ упознати најважнија функција у репродуковању слике на *LED* дисплеју представља *refreshDisplay()*, којој је потребно неколико *ms* (тачније око ~80*ms*) за њено извршавање. Што значи да се током 1 секунде, слика прикаже приближно 12 пута, или 12*fps* (енг. *frame per second*). Не постоји тачан број колико људско око може уочити слика у секунди. Просек је око 30-35*fps*, али постоје и изузеци. Наравно, уочавање брзине смењивања слике зависи и од њихове међусобне сличности. Па ипак, број који човек може уочити и обрадити креће се тек око 24 до 28*fps* [4]. Са реализацијом у *Python*-у, та вредност је дупло већа од наше, и из тог разлога ми видимо "треперење" на дисплеју. Зато ће функција *refreshDisplay()* бити реализована у *C*, спакована у *Python* библиотеку и позивана из *Python* апликације.

У функцији *refreshDisplay()* се врши манипулација излазним *GPIO* пиновима. Док се за реализацију у *Python*-у користила библиотека *RPi.GPIO* за манипулацију *GPIO* пиновима, за реализацију у *C* ће се користити библиотека *bcm2835*. Ова библиотека омогућава приступ *GPIO* и другим *IO* функцијама на *Broadcom BCM* 2835 чипу. Иако се на *Raspberry Pi* 4 налази чип *Broadcom BCM* 2711, ова библиотека је компатибилна и са тим чипом. Она обезбеђује функције за читање дигиталног улаза и сетовање дигиталног излаза, користећи *SPI* и *I2C* комуникацију за приступ системским бројачима. Детекција догађаја на пину је омогућена помоћу *polling*, док прекиди (енг. *interrupts*) нису могући. Ова библиотека ради на свим врстама *Debian* оперативног система. Такође је компатибилна са *C++*. Инсталира се као *header* фајл и недељена библиотека на било ком *LINUX* репозиторијуму, али наравно њена примена је могућа само на уређајима са *BCM2835* чипом и чиповима који су компатибилни са њим.

Language	Library	Tested / version	Square wave
Shell	/proc/mem access	2015-02-14	2.8 kHz
Shell / gpio utility	WiringPi gpio utility	2015-02-15 / 2.25	40 Hz
Python	RPi.GPIO	2015-02-15 / 0.5.10	70 kHz
Python	wiringpi2 bindings	2015-02-15 / latest github	28 kHz
Ruby	wiringpi bindings	2015-02-15 / latest gem (1.1.0)	21 kHz
C	Native library	2015-02-15 / latest RaspPi wiki code	22 MHz
C	BCM 2835	2015-02-15 / 1.38	5.4 MHz
C	wiringPi	2015-02-15 / 2.25	4.1 – 4.6 MHz
Perl	BCM 2835	2015-02-15 / 1.9	48 kHz

Слика 5.*Упоређивање брзине приступа GPIO пиновима различитих библиотека* [5]

Када је цела функција имплементирана у *C*, потребно је искомпајлирати и генерисати *.so* фајл (тај фајл представља дељени објектни фајл за дељену библиотеку). Да бисмо користили функције потребно је укључити *Python* пакет *ctypes*. Он представља библиотеку која обезбеђује *C* компатибилне типове података и позивање функција из *DLL* или дељених бибиотека. Покретањем команде *CDLL* из пакета *ctypes* врши се преузимање динамичког линка библиотека. Цикличним позивањем функције *refreshDisplay*(), добијамо изглед дисплеја:

Слика 6. *Испис садржаја на дисплеју помоћу C апликације*

И коначно време потребно за извршавање функције *refreshDisplay()* износи у просеку ~4.5ms, што значи да број слика или освежавања током 1s износи ~220*fps*, што је много више од вредности које људско око може да детектује. Оваквом реализацијом, „треперење" се више не појављује на дисплеју.

```
0.004473209381103516
0.004437923431396484
0.005208730697631836
0.005080223083496094
0.005780935287475586
0.00539708137512207
0.004472017288208008
0.004460334777832031
0.004877567291259766
0.0047566890716552734
```

Слика 7. *10 узорака мерења времена извршавања функције refreshDisplay() са реализацијом у C*

5. ЗАКЉУЧАК

У претходним поглављима смо се више упознали са дисплејима са пасивном матрицом и доказали да без обзира колико је *Python* погодан за различите апликације, и даље је *C* бољи избор за временски критичне функције. *Python* није погодан за коришћење у *embedded* системима за извршавање временски критичних задатака. Ово би још више дошло до изражаја да смо користили веће дисплеје.

Оно што се исто може приметити са Слике 5. је да постоји простор за даље убрзавање приступа *GPIO* пиновима, уместо коришћења неке *GPIO* библиотеке, директним приступом регистрима. Један од корака за убрзавање извршавања наше *refreshDisplay*() функције би био управо тај.

6. ЛИТЕРАТУРА

[1] J. C. Бајић, "*LED* дисплеј", https://www.optolab.ftn.uns.ac.rs/images/NASTAVA/OE/files/lab/pdf/2021/11-LED-panel.pdf; [Приступљено у октобру 2022.]

[2] http://www.racunarstvo.matf.bg.ac.rs/MasterRadovi/2014_03_10_Ksenija_Grujic-Petrovic/rad.pdf; [Приступљено у октобру 2022.]

[3] Mike Huls, "Why Python is so slow and how to speed it up", https://towardsdatascience.com/why-is-python-so-slow-and-how-to-speed-it-up-485b5a84154e [Приступљено у октобру 2022.]

[4] https://www.healthline.com/health/human-eye-fps [Приступљено у октобру 2022.]

[5] https://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/ [Приступљено у октобру 2022.]

Кратка биографија:

Стефан Новаковић рођен је 23.08.1995. у Новом Саду. Основну школу "Петар Кочић" у Темерину завршио је 2010. године. Гимназију "Светозар Марковић" у Новом Саду завршио је 2014. године. Исте године уписује Факултет техничких наука, смер Енергетика, електроника и телекомуникације. У октобру 2019. године завршава основне студије, након чега уписује мастер студије такође на Факултету Техничких Наука, смер Примењена електроника.