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

**UNIVERZITET U NOVOM SADU**

**FAKULTET TEHNIČKIH NAUKA**

**NOVI SAD**

**Departman za računartvo i automatiku**

**Odsek za računarsku tehniku i računarske komunikacije**

**ISPITNI RAD**

**Kandidat: Filip Jašić , Ana Teodorović**

**Број индекса: RA46/2014 , RA191/2016**

**Predmet: Sistemska programska podrška u realnom vremenu II**

**Tema rada: Sajmon igra za jednog igrača**

**Mentor rada: prof. Dr Miroslav Popović**

**Novi Sad, januar, 2019**

**Sadržaj**

1. **Uvod**
   1. **Sajmon igra**
   2. **Zadatak**
2. **Analiza problema**
3. **Koncept rešenja**
   1. **Kernel modul**
   2. **Testna aplikacija**
4. **Opis rešenja**
   1. **Funkcija hrtimer\_restart**
   2. **Funkcija irqreturn\_t h\_irq\_gpio3**
   3. **Funkcija irqreturn\_t h\_irq\_gpi22**
   4. **Funkcija gpio\_driver\_init**
   5. **Funkcija gpio\_driver\_exit**
   6. **Funkcija gpio\_driver\_read**
   7. **Funkcija gpio\_driver\_write**
5. **Testiranje**
6. **Zaključak**

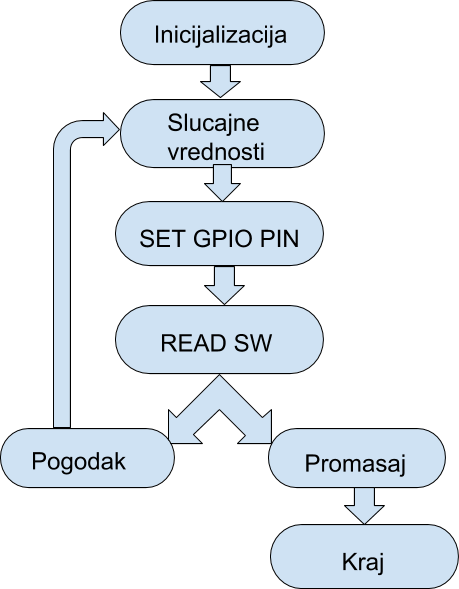
**1. Uvod**

**1. Sajmon igra**

Sajmon je igra memorijskih veština. Počinje tako što uređaj zadaje određene boje u određenom redosledu, a zadatak igrača je da ponove niz. Ukoliko je igrač uspeo, niz se povećava i igra postaje komplikovanija. Ukoliko igrač poreši, igra se završava.

**2. Zadatak**

Partija počinje tako što se pali određen broj dioda na nasumičnim lokacijama. U zavisnosti od toga da li je igrač pogrešio i nije ugasio prekidače u pravilnom redosledu i na korespodentnim lokacijama gde se nalaze diode, igra se završava i ispisuje se rezultat. Ukoliko igrač nije pogrešio, igra se nastavlja i broj upaljenih dioda se povećava za 1.



**2. Analiza problema**

Koriscena je logika da se detektuje promena u statusu switcha,posto ako bi koristili drugaciji nacin,odnosno da ako bi nakon svakog koriscenja switcha vracali ga u pocetno stanje moglo bi da dodje do nepozeljne promene u statusu,ili da se promena stanja switcha uopste ne detektuje.Prvo smo probali da dodajemo delay izmedju svake promene,ali to nije bilo dovoljno kada se broj upaljenih lampica uveca.A jos gora situacija je bila kada bi se lampica upalila na istom mestu izmedju kratkih intervala.Zbog svega ovoga navedenog,koriscena je gorenavedena logika.

**3. Koncept rešenja**

**1. Kernel modul**

Potrebno je realizovati rukovalac kao karakterni uredjaj,i testna aplikacija treba da proveri njegovu funkcionalnost.Testna aplikacija se povezuje sa rukovaocem na osnovu **major** broja koji naš rukovaoac ispisuje na standardni izlaz upotrebom komande dmesg(ili sudo cat /dev/kmsg za konstantni ispis).

Na osnovu fizičke arhitekture LE dioda(LED0-LED3) i prekidača(SW-SW3),odnosno njihovog mapiranja na odgovarajuće GPIO prolaze date na slici 1,potrebno je:

1. Implementirati podršku za rukovanje ulazno/izlaznim ureђaјima (GPIO).Rukovalac GPIO treba da omogući:

* U postupku iniciјalizaciјe modula postaviti GPIO prolaze vezane na diode/prekidače u odgovaraјući smer (GPIO za LE diode kao izlazne, GPIO za prekidače kao ulazne),
* Promenu stanja LE dioda kroz operaciјu pisanja:
  + LED x 1 (uključivanje LE diode x)
  + LED x 0 (isključivanje LE diode x)
* Prilikom implementaciјe voditi se primerom sa slike, odnosno koristiti date identifikatore prolaza i koristiti postoјeće funkciјe za definisanjesmera prolaza, promenu stanja izlaza (diode), odnosno čitanje vrednosti sa ulaza (prekidači):
  + void SetGpioPinDirection(char pin, char direction)
  + void SetGpioPin(char pin)
  + void ClearGpioPin(char pin)
  + char GetGpioPinValue(char pin)

2. Napraviti štopericu sa preciznošću od 1s.

* void hrtimer\_init(struct hrtimer \*timer, clockid\_t clock\_id, enum hrtimer\_mode mode)
* int hrtimer\_start(struct hrtimer \*timer, ktime\_t tim, const enum hrtimer\_mode mode)
* static inline ktime\_t ktime\_set(const s64 secs, const unsigned long nsecs)
* int hrtime\_cancel(struct hrtimer \*timer)
* u64 hrtimer\_forward(struct hrtimer \*timer, ktime\_t now, ktime\_t interval)
* void gpio\_free(unsigned gpio)

3. Proširiti rukovalac tako da se omogući čitanje stanja sa prekidača kroz operaciјu čitanja.

4. Napraviti pseudoslučajni generator brojeva,koji odeđuje koje će diode biti uplaljene i čiji broj se uvećava za 1 sa svakom “dobrom” iteracijom.

5.Podesiti prekide tako da se aktiviraјu na opdaјuću ivicu tastera PB0 i PB1. Pritisak na taster PB0 pokreće igru ukoliko niјe pokrenuta ili јe zaustavljena, a taster PB1 zaustavlja pokrenutu igru, a resetuјe zaustavljenu igru.

**2. Testna aplikacija**

Potrebno je proširiti test aplikaciјu upotrebom pthread biblioteke tako da se omogući konkuretno čitanje promene u rezultatu.Napraviti 1 nit koјe da li je korisnik napravio gresku i ako jeste da se rezultat ispise u korisnicki prostor.

Samo pokretanje testne aplikacije se vrši pozicioniranjem u **gpio\_driver\_test\_app/bin/Release/** i pozivanjem sa **./gpio\_driver\_test\_app.** Ova aplikacija čita podatke iz prethodno napravljenog karakternog uređaja koji se nalazi na lokaciji **/dev/gpio\_driver** i ispisuje ih na standardni izlaz.



 Slika 1 Mapiranje LE dioda i prekidača na GPIO prolaze

**4. Opis rešenja**

1.Funkcija hrtimer\_restart

Poziva se kada na štoperici istekne vreme u trajanju od 1 sekunde.

Ukoliko je pritisnut taster za pokretanje igre, pokreće se štoperica koja se povećava nakon svake iteracije, odnosno određiaće broj upaljenih dioda.

Kada dodje do greške, zabeleži se broj pogodaka, i igra se vraća na početak.

2. Funkcija irqreturn\_t h\_irq\_gpio3

Poziva se kada je pritisnut taster za pokretanje igre(PB0).

3.Funkcija irqreturn\_t h\_irq\_gpio22

Poziva se kada je pritisnut taster za reset(PB1).

4. Funkcija gpio\_driver\_init

Registruje upravljački program uređaja, dodeljuje bafer, inicijalizuje bafer, mapira GPIO fizičke adrese na virtuelne adrese, inicijalizuje GPIO pinove i pokreće tajmer.

5.Funkcija gpio\_driver\_exit

Zaustavlja tajmer, oslobađa GPIO pinove, oslobađa bafer,demapira GPIO fizičke adrese sa virtualnih adresa, odjavljuje upravljački program uređaja.

6.Funkcija gpio\_driver\_read

Prenosi podatke iz bafera upravljačkog uređaja do korisnika primenom funkcije copy\_to\_user.

7.Funkcija gpio\_driver\_write

Prenosi podatke od korisničkog prostora do kernel prostora.

**4. Testiranje**

Testiranje je bilo fokusirano na tome da drajver upesno prepozna promenu stanja prekidaca i to pravovremeno.Na primer da li ce ,pre nego sto je zavrsena sekvenca paljenja LED dioda a vec se pocelo sa paljenjem prekidaca ,se oni i uspesno registrovati.To se pokazalo uspesnim.