

Област вежби: *Linux* руковаоци

РУКОВАОЦИ У/И УРЕЂАЈА НА LINUX ОПЕРАТИВНОМ СИСТЕМУ – ЗАДАТАК 2

Предуслови:

- Rpi2 рачунар са додатном плочицом са LE диодама, прекидачима и тастерима,
- Преводиоц *GCC* освежен на верзију 4.7 или новију,
- Преузето и подешено језгро Linux оперативног система на Raspberry Pi уређају, према опису из документа „УВОД - Raspberry Pi рачунар“,
- Подешен мрежни приступ на један од начина представљених у документу „УВОД - Raspberry Pi рачунар“ уколико се ради преко мреже. Ако се Rpi2 рачунар користи као самосталан рачунар овај захтев се може занемарити,
- Познавање језика Це и материјала из вежби „УВОД У КОНКУРЕНТНО ПРОГРАМИРАЊЕ, „СИНХРОНИЗАЦИЈА И СИГНАЛИЗАЦИЈА ПРОГРАМСКИХ НИТИ“ и „ТУТОРИЈАЛ О РУКОВАОЦИМА У/И УРЕЂАЈА НА LINUX ОПЕРАТИВНОМ СИСТЕМУ“.

Увод

У овој вежби приложен је код за Linux модул и пример тестне апликације у препознатљивој структури директоријума са одговарајућим Makefile-овима. Да би се реализовао задатак, неопходно је проширити дати Linux модул и направити одговарајућу тестну апликацију, а приложену користити као полазну тачку. Приложени Linux модул у прекидној рутини за прекид везан на тастер PB0 преузима вредност прекидача SW0 и исписује га у *kernel log*. Прекид се активира на опадајућу ивицу, односно на отпуштање тастера с обзиром да се притиском тастера добија логичка 1, а отпуштањем логичка 0.

Задатак

Задатак је потребно реализовати по угледу на примере из туторијала о прављењу руковаоца („ТУТОРИЈАЛ О РУКОВАОЦИМА У/И УРЕЂАЈА НА LINUX ОПЕРАТИВНОМ СИСТЕМУ“) и конкурентног програмирања („УВОД У КОНКУРЕНТНО ПРОГРАМИРАЊЕ“ и „СИНХРОНИЗАЦИЈА И СИГНАЛИЗАЦИЈА ПРОГРАМСКИХ НИТИ“). Руковалац *memory* представља репрезентативни пример на основу којег треба да се реализује руковалац и његова

тестна апликација која треба да поровери његову функционалност. И овај руковалац треба реализовати као карактерни уређај (chardev).

На основу физичке архитектуре LE диода (LED0~LED3), прекидача (SW0~SW3) и тастера (PB0 и PB1), односно њиховог мапирања на одговарајуће GPIO пролазе дате на слици 1, потребно је:

1. Имплементирати руковалац GPIO, који има функционалност штоперице, кроз подршку за руковање улазно/излазним уређајима (GPIO). Руковалац GPIO треба да омогући:
 - У поступку иницијализације модула поставити GPIO пролазе везане на диоде/прекидаче/тастере у одговарајући смер (GPIO за LE диоде као излазне, GPIO за прекидаче и тастере као улазне),
 - Користећи временску контролу, по угледу на претходну вежбу, реализовати штоперицу са прецизношћу од 1s, а тренутно стање у бинарном облику (само најнижа 4 бита) представити на 4 доступне LE диоде. Као и у претходној вежби, користити следеће функције за покретање/заустављање временске контроле:

```
void hrtimer_init(struct hrtimer *timer, clockid_t clock_id, enum hrtimer_mode mode)
```

Функција	Опис
<i>hrtimer_init</i>	иницијализује временску контролу за дати такт
Параметри	Опис
<i>timer</i>	временска контрола која се иницијализује. Структура чије је најважније поље <code>function</code> - адреса функције временске контроле која се декларише као: <code>enum hrtimer_restart function(struct hrtimer *param);</code>
<i>clock_id</i>	такт који се користи
<i>mode</i>	начин дефинисања времена истека, апсолутно време (HRTIMER_ABS) или релативно (HRTIMER_REL)

```
int hrtimer_start(struct hrtimer *timer, ktime_t tim, const enum hrtimer_mode mode)
```

Функција	Опис
<i>hrtimer_start</i>	(поново) покреће временску контролу процесора
Параметри	Опис
<i>timer</i>	временска контрола која се додаје
<i>tim</i>	време истека временске контроле
<i>mode</i>	начин дефинисања времена истека, апсолутно време (HRTIMER_MODE_ABS) или релативно (HRTIMER_MODE_REL)
Повратна вредност	Опис
<i>int</i>	Уколико је покретање временске контроле успело повратна вредност је 0. У супротном, уколико је временска контрола већ покренута, повратна вредност је 1.

```
static inline ktime_t ktime_set(const s64 secs, const unsigned long nsecs)
```

Функција	Опис
<i>ktime_set</i>	поставља ktime_t променљиву на основу датог броја секунди и наносекунди
Параметри	Опис
<i>secs</i>	број секунди
<i>nsecs</i>	број наносекунди
Повратна вредност	Опис
<i>ktime_t</i>	Структура попуњена датим вредностима секунди и наносекунди

```
int hrtimer_cancel(struct hrtimer *timer)
```

Функција	Опис
<i>hrtimer_cancel</i>	поништава временску контролу и чека на завршетак
Параметри	Опис
<i>timer</i>	адреса временске контроле која се прекида
Повратна вредност	Опис
<i>int</i>	0 – временска контрола није била активна 1 – временска контрола је била активна

```
u64 hrtimer_forward(struct hrtimer *timer, ktime_t now, ktime_t interval);
```

Функција	Опис
<i>hrtimer_forward</i>	продужава (понавља) истек временске контроле
Параметри	Опис
<i>timer</i>	временска контрола која се продужава
<i>now</i>	продужава се у односу на тренутно време
<i>interval</i>	интервал за који се продужава
Повратна вредност	Опис
<i>u64</i>	број прекорачења времена

Приликом имплементације водити се примером са слике, односно користити дате идентификаторе пролаза и користити постојеће функције за дефинисање смера пролаза, промену стања излаза (диоде), односно читање вредности са улаза (прекидачи и тастери):

- void SetGpioPinDirection(char pin, char direction)
- void SetGpioPin(char pin)
- void ClearGpioPin(char pin)
- char GetGpioPinValue(char pin)

2. Водећи се датим примером подешавања прекида и примером прекидне рутине, подесити прекиде тако да се активирају на опдајућу ивицу тастера PB0 и PB1. Притисак на тастер PB0 покреће штоперицу уколико није покренута или је заустављена, а тастер PB1 зауставља покренуту штоперицу, а ресетује заустављену штоперицу. Као и у примеру, користити следеће функције за иницијализацију/уклањање прекидне рутине:

```
int gpio_request_one(unsigned gpio, unsigned long flags, const char
*label);
```

Функција	Опис
<i>gpio_request_one</i>	захтева један GPIO са почетном конфигурацијом
Параметри	Опис
<i>gpio</i>	GPIO број
<i>flags</i>	GPIO конфигурација дефинисана помоћу GPIOF *
<i>label</i>	описни стринг за GPIO
Повратна вредност	Опис
<i>int</i>	0 уколико је успешно, код грешке у супротном

```
void gpio_free(unsigned gpio);
```

Функција	Опис
<i>gpio_free</i>	ослобађа GPIO
Параметри	Опис
<i>gpio</i>	GPIO број

```
int gpio_to_irq(unsigned gpio);
```

Функција	Опис
<i>gpio_to_irq</i>	враћа IRQ који одговара датом GPIO
Параметри	Опис
<i>gpio</i>	GPIO чији ће IRQ бити прочитан
Повратна вредност	Опис
<i>int</i>	IRQ број одговарајући за дати GPIO или код грешке у случају исте.

```
int request_irq(unsigned int irq, irq_handler_t handler, unsigned long
flags, const char *name, void *dev)
```

Функција	Опис
<i>request_irq</i>	заузимање линије прекида
Параметри	Опис
<i>irq</i>	линија прекида која се заузима
<i>handler</i>	функција која ће бити позвана из нити прекидне рутине. Декларише се као: static irqreturn_t function(int irq, void *data)
<i>flags</i>	IRQF_SHARED - дељени прекид IRQF_TRIGGER * - дефиниција активне ивице или нивоа
<i>name</i>	стринг са именом уређаја који заузима прекидну линију
<i>dev</i>	аргумент који ће бити прослеђен прекидној рутини, најчешће служи за прослеђивање адресе структуре која представља физички уређај
Повратна вредност	Опис
<i>int</i>	број прекорачења времена

```
void disable_irq(unsigned int irq);
```

Функција	Опис
<i>disable_irq</i>	искључује IRQ и чека на завршетак операције
Параметри	Опис
<i>irq</i>	линија прекида која се искључује

```
void free_irq(unsigned int irq, void *dev_id);
```

Функција	Опис
<i>free_irq</i>	уклања прекидну рутину. Чека на завршетак операције
Параметри	Опис
<i>irq</i>	линија прекида за коју се уклања прекидна рутина
<i>dev_id</i>	идентификатор уређаја који уклања прекидну рутину

3. Проширити руковалац и тест апликацију тако да се омогући читање времена штоперице из руковаоца и исписује на екран.

LED0	LED1	LED2	LED3	SW0	SW1	SW2	SW3	PB0	PB1
GPIO6	GPIO13	GPIO19	GPIO26	GPIO12	GPIO16	GPIO20	GPIO21	GPIO3	GPIO22

Слика 1 Мапирање LE диода, прекидача и тастера на GPIO пролазе