



# UNIVERZITET U NOVOM SADU FAKULTET TEHNIČKIH NAUKA



**UNIVERZITET U NOVOM SADU  
FAKULTET TEHNIČKIH NAUKA  
NOVI SAD**

**Departman za računarstvo i automatiku**

**Odsek za računarsku tehniku i računarske komunikacije**

## PROJEKTNI ZADATAK

**Kandidati:** Mladen Bazina RA194/2020  
Ivan Sauer RA202/2018

**Predmet:** Operativni sistemi za rad u realnom vremenu  
**Tema rada:** servo\_fb

**Mentor rada:** dr Miloš Subotić  
prof. dr Miroslav Popović  
TA Andrej Gobor

17. Januar. 2024.

## Sadržaj:

1. Uvod.....	2
1.1. Zadatak projekta.....	2
1.2. Ciljevi projekta.....	2
2. Isključivanje „lažne“ povratne sprege.....	2
3. Implementacija ISR funkcije.....	3
4. Implementacija funkcije koja računa faktor ispune.....	3
5. Pisanje test programa za testiranje rada povratne sprege.....	4
6. Komande za buildovanje i pokretanje programa.....	5
7. Zaključak.....	5

---

# 1. Uvod

## 1.1 Zadatak projekta

Naziv izrađenog projekta je servo\_fb. Preciznije rečeno projekat se odnosi na rad sa servo motorom i sa određivanjem položaja servo motora pomoću povratne sprege.

## 1.2 Ciljevi projekta

Ciljevi izrade ovog projekta su:

- Isključiti „lažnu“ povratnu spregu.
- Implementirati funkciju koja će da sačuva vremenski momenat opadajuće i rastuće ivice PWM signala.
- Implementirati funkciju koja će date vremenske trenutke da obradi i da nam izračuna faktor ispune PWM signala.
- Napisati testnu aplikaciju koja će da ispita valjanost samog koda.

## 2. Isključivanje "lažne" povratne sprege

Prostom promenom vrednosti sa 1 na 0 već definisane konstante FAKE\_FEEDBACK usmeravamo program da očitava faktor ispune aktivacionog PWM signala servo motora.

```
#define FAKE_FEEDBACK 0

#if FAKE_FEEDBACK
    a.pos_fb[ch] = pos_cmd[ch]; // Loop pos cmd back.
#else
    //TODO test
    servo_fb__get_pos_fb(ch, &a.pos_fb[ch]);
#endif
```

### 3. Implementacija ISR funkcije

Potrebno je bilo implementirati funkciju koja će da se poziva na svaku promenu stanja na određenom GPIO pinu. Zatim treba da ažurira vremenske podatke i vrši određene operacije u skladu sa time da li je detektovala rastuću ili opadajuću ivicu signala.

```
static irqreturn_t fb_isr(int irq, void* data) {
    u64 t = ktime_get_ns();
    servo_fb_t* p = (servo_fb_t*)data;
    bool s = gpio__read(p->pin);
    if(s){
        // It was rising edge.
        atomic64_set(&p->T_on, p->t_fe);
    }else{
        // It was falling edge.
        p->t_fe = t;
    }
    //TODO calc duty
    return IRQ_HANDLED;
}
```

### 4. Implementacija funkcije koja računa faktor ispunje signala

Ova funkcija ima za cilj pružiti informacije o trenutnom stanju servo motora. Ona koristi informacije koje pruža funkcija fb\_isr kako bi izračunala faktor ispunje signala. Faktor ispunje koristi se kako bi znali u kom položaju (pod kojim uglom) se nalazi servo motor.

```
#define INT64_T_ONE 1ULL
#define DIV_SHIFT 32
#define DIV_MUL ((INT64_T_ONE<<DIV_SHIFT)/20000)
#define DIV_ADD (INT64_T_ONE<<(DIV_SHIFT-1))

void servo_fb__get_pos_fb(servo_fb__ch_t ch, u16* pos_fb) {
    u64 T_on; // [ns]
    u16 duty; // [permille]

    // Make local copy of measurement.
    T_on = atomic64_read(&servo_fb[ch].T_on);

    // Hard-coded for 50 Hz.
    // duty = T_on/20000;
    duty = (T_on*DIV_MUL + DIV_ADD) >> DIV_SHIFT;

    printk("T_on: %lld, duty: %d", T_on, duty);

    *pos_fb = duty;
}
```

## 5. Pisanje test programa za testiranje rada povratne sprege

Program test\_fb ima funkcionalnost testiranja i kontrole motora. Funkcija parse\_args parsira argumente komandne linije kako bi dobio informacije o indeksu servoa i dužini impulsa. Ako je broj argumenata jednak 2, proverava da li je prvi argument -h ili --help i ispisuje informacije o korišćenju. Ako je broj argumenata jednak 3, pokušava da konvertuje argumente u celobrojne vrednosti i postavlja vrednosti p\_servo\_idx i p\_angle. Unutar main funkcije otvara se uređajski fajl (DEV\_FN) pomoću funkcije open u read/write režimu. Komunikacija sa uređajem izvršava se putem file descriptor-a kroz operacije read i write. Dužnosti motora se čuvaju u nizu duties, koji se čita i piše u uređajski fajl. Kroz komandnu liniju, korisnik može postaviti indeks servo motora i vrednost dužnosti tog servo motora. Program prikazuje informacije o dužnosti pre i posle pisanja/čitanja iz uređajskog fajla.

```
fd = open(DEV_FN, O_RDWR);
if(fd < 0){
    fprintf(stderr, "ERROR: \"%s\" not opened!\n", DEV_FN);
    fprintf(stderr, "fd = %d %s\n", fd, strerror(-fd));
    return 4;
}

printf("duty = %d\n", duty);
duties[servo_idx] = duty; // [permilles]

for(int i = 0; i < MOTOR_CLTR__N_SERVO; i++){
    printf("duties[%d] = %d\n", i, duties[i]);
}

r = write(fd, (char*)&duties, sizeof(duties));
if(r != sizeof(duties)){
    fprintf(stderr, "ERROR: write went wrong!\n");
    return 4;
}

|

r = read(fd, (char*)&duties, sizeof(duties));

if(r != sizeof(duties)){
    fprintf(stderr, "ERROR: read went wrong!\n");
    return 4;
}
for(int i = 0; i < MOTOR_CLTR__N_SERVO; i++){
    printf("duties[%d] = %d\n", i, duties[i]);
}

duty = duties[servo_idx]; // [permilles]
printf("duty = %d\n", duty);
```

## 6. Komande za buildovanje i pokretanje programa

Da bi buildovali program prvo je potrebno pozicionirati se u ROS/arm\_and\_chassis\_ws direktorijum.

Build se vrši pomoću skripte tmuxer.py. Nju pozivamo tako što u terminal upišemo

```
./tmuxer.py build
```

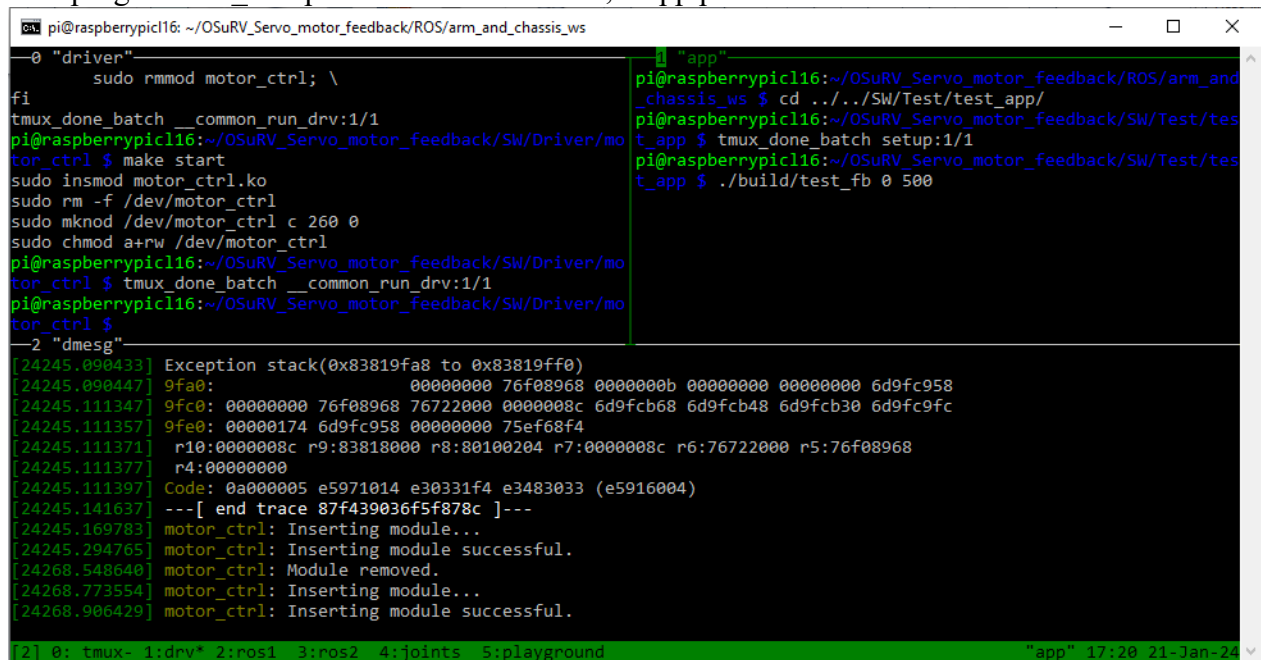
Pokretanje:

```
./tmuxer.py run
```

Kada pokrenemo tmuxer.py, otvoriće nam se terminalski multiplekser. U donjem levom uglu ćemo videti nekoliko kartica (tabova)

```
[2] 0: tmux- 1:drv* 2:ros1 3:ros2 4:joints 5:playground
```

Test program test\_fb.c pokreće se u drv kartici, u app prozoru.



```
pi@raspberrypic16: ~/OSuRV_Servo_motor_feedback/ROS/arm_and_chassis_ws
-0 "driver"
  sudo rmmod motor_ctrl; \
fi
tmux_done_batch __common_run_drv:1/1
pi@raspberrypic16:~/OSuRV_Servo_motor_feedback/SW/Driver/motor_ctrl $ make start
sudo insmod motor_ctrl.ko
sudo rm -f /dev/motor_ctrl
sudo mknod /dev/motor_ctrl c 260 0
sudo chmod a+rw /dev/motor_ctrl
pi@raspberrypic16:~/OSuRV_Servo_motor_feedback/SW/Driver/motor_ctrl $ tmux_done_batch __common_run_drv:1/1
pi@raspberrypic16:~/OSuRV_Servo_motor_feedback/SW/Driver/motor_ctrl $
-2 "dmesg"
[24245.090433] Exception stack(0x83819fa8 to 0x83819ff0)
[24245.090447] 9fa0: 00000000 76f08968 0000000b 00000000 00000000 6d9fc958
[24245.111347] 9fc0: 00000000 76f08968 76722000 0000008c 6d9fcb68 6d9fcb30 6d9fc9fc
[24245.111357] 9fe0: 00000174 6d9fc958 00000000 75ef68f4
[24245.111371] r10:0000008c r9:83818000 r8:80100204 r7:0000008c r6:76722000 r5:76f08968
[24245.111377] r4:00000000
[24245.111397] Code: 0a000005 e5971014 e30331f4 e3483033 (e5916004)
---[ end trace 87f439036f5f878c ]---
[24245.169783] motor_ctrl: Inserting module...
[24245.294765] motor_ctrl: Inserting module successful.
[24268.548640] motor_ctrl: Module removed.
[24268.773554] motor_ctrl: Inserting module...
[24268.906429] motor_ctrl: Inserting module successful.
[2] 0: tmux- 1:drv* 2:ros1 3:ros2 4:joints 5:playground "app" 17:20 21-Jan-24
```

Kada želimo da izađemo iz TMux-a potrebno je da odemo u tmux karticu I stisnemo enter.

## 7. Zaključak

Ovaj Linux kernel modul pruža podršku za upravljanje servo i BLDC motorima kroz korišćenje PWM signala. Kroz karakteristični fajl korisnici mogu postavljati željene pozicije servo motora, a modul omogućava čitanje informacija o trenutnom položaju servo motora i broju pulsacija BLDC motora. Implementacija koristi različite module za upravljanje hardverskim resursima poput GPIO pinova i PWM signala.