KATEDRA ZA ELEKTRONIKU

SISTEMI U REALNOM VREMENU

HARIS TURKMANOVIĆ - 2024/25

FreeRTOS

Taskovi

OSOBINE TASKOVA KARAKTERISTIČNE ZA FREERTOS

FREERTOS TASK API

PRIMERI

Osobine

- Svakom kreiranom tasku je moguće dodeliti prioritet
  - Opseg prioriteta taskova se kreće od 0 do (`configMAX_PRIORITIES – 1`)
    - `configMAX_PRIORITIES` je makro koji je moguće menjati u okviru konfiguracije ***FreeRTOS***-a
- Funkcionalnosti *FreeRTOS* taska su implementirane u okviru klasične C-ovske funkcije
  - Na dalje ćemo ovu funkciju zvati „Task f-ja“
  - Task f-ja mora imati sledeći potpis (deklaraciju)

```
void vTaskFunctionName(void *pvParameter);
```

- Osobine Task f-je su:
  - U funkciju se ulazi samo jednom
    - Funkciju poziva scheduler nakon startovanja
  - Funkcionalnost taska se implementira u okviru beskonačne petlje
  - Iz Task f-je se ne izlazi eksplicitno (klasičnim pozivom ***return*** naredbe)
    - Ukoliko je potrebno izaći iz funkcije (task više nije potreban) potrebno ga je izbrisati
  - Jedna Task f-ja se može koristi kao izvršna funkcija više taskova
    - U tom slučaju **svaki kreirani task** će imati svoju **posebnu** instancu Task f-je koja će imati svoj zaseban stek
    - Na primer, TASK A i TASK B implementiraju isti algoritam, ali nad različitim parametrima i treba da se izvršavaju paralelno. Ovakav slučaj, na primer, može da se desi u slučaju paralelnog strimovanja podataka.

Osobine

- Svakom kreiranom tasku je moguće dodeliti prioritet
  - Opseg prioriteta taskova se kreće od 0 do (`configMAX_PRIORITIES – 1`)
    - `configMAX_PRIORITIES` je makro koji je moguće menjati u okviru konfiguracije ***FreeRTOS***-a
- Funkcionalnosti *FreeRTOS* taska su implementirane u okviru klasične C-ovske funkcije
  - Na dalje ćemo ovu funkciju zvati „Task f-ja“
  - Task f-ja mora imati sledeći potpis (deklaraciju)

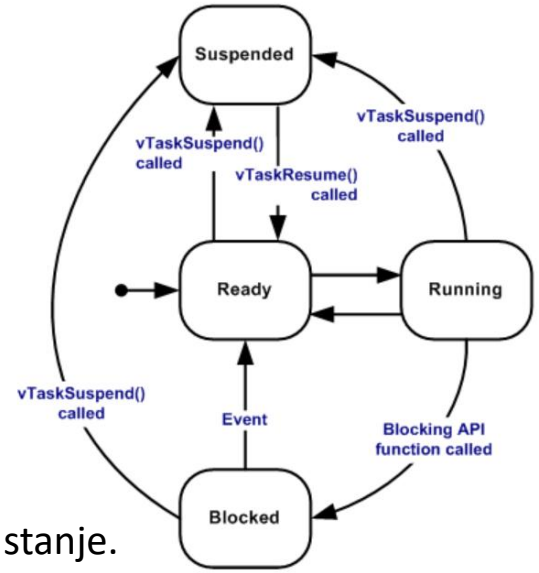
```
void vTaskFunctionName(void *pvParameter);
```

- Osobine Task f-je su:
  - Struktura koda jedne tipične Task f-je u *FreeRTOS*-u

```
void ATaskFunction( void *pvParameters )
{
    /*  Declare variables which will be used only by this task
    *   Variables are declared on ordinary way
    */
    int32_t lVariableExample = 0;
    /* A task will normally be implemented as an infinite loop. */
    for( ;; )
    {
        /* The code to implement the task functionality will go here. */
    }
    /* Should the task implementation ever break out of the above loop, then the task
    must be deleted before reaching the end of its implementing function. The NULL
    parameter passed to the vTaskDelete() API function indicates that the task to be
    deleted is the calling (this) task.  */
    vTaskDelete( NULL );
}
```

Osobine

- Task može biti u jednom od sledećih stanja
  - **Running** – Task se trenutno izvršava
    - Na platformi koja ima jedno procesorsko jezgro samo jedan task u sistemu može biti u ovom stanju
  - **Ready** – Task je u listi taskova spremnih za izvršavanje
    - Task nije u stanju *Blocked* ili Suspended ali se trenutno ne izvršava jer postoji drugi task jednakog ili višeg prioriteta koji se trenutno nalazi u stanju *Running*
  - **Blocked** – Task je blokiran i čeka generisanje nekog događaja
    - Task se može blokirati čekajući na **semaforu**, **grupi događaja**, **queue** ili **notifikaciji** (sve ove objekte ćemo detaljno obraditi u nastavku. Za sada je bitno da znamo šta uzrokuje blokiranje taska)
    - U okviru FreeRTOS-a task koji je u ovom stanju ima **timeout** period. Nakon isteka ovog vremena, tesk će biti odblokiran čak i ako događaj na koji task čeka nije generisan.
      - „Beskonačno blokiranje“ se može realizovati tako što će timeout biti „beskonačan“



  - **Suspended**
    - Task ulazi u ovo stanje isključivo eksplicitnim pozivom API funkcija koje task stavljaju o ovo stanje.
    - Task koji se nalazi u ovom stanju ne učestvuje u raspoređivanju od strane *Schedulera*
    - Task izlazi iz ovog stanja samo eksplicitnim pozivom API funkcija koje vraćaju task u stanje *Ready*

API funkcije

- Neke od **najčešće** korišćenih API f-ja *FreeRTOS*-a koje su neophodne za rad sa taskovima

Funkcija	Opis
`xTaskCreate`	*Kreira Task*
`vTaskDelete`	*Briše Task*
`vTaskDelay`	*Blokira **pozivajući** task određeni period vremena*

- **Detaljniji** opisi **svih API funkcija** za rad sa taskovima dostupni su u okviru zvanične FreeRTOS dokumentacije (slajd 27)
- Dodatne API funkcije koje nam budu trebale za rešavanje primera ćemo uvoditi i kroz ove prezentacije

Kreiranje taskova

- Kreiranje taskova se vrši koristeći `xTaskCreate` funkciju koja ima sledeću deklaraciju

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        uint16_t usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );
```

Naziv parametra	Opis
pvTaskCode	Prethodno kreirana task f-ja
pcName	Naziv taska
usStackDepth	Veličina steka koja se dodeljuje tasku. Jedinica je širina procesorske reči u bajtovima!
pvParameters	Parametar koji prilikom pokretanja task želimo da prosledimo tasku
uxPriority	Prioritet taska
pxCreatedTask	Instanca kreiranog taska (implicitno predstavlja pokazivač na kreirani task)

- Ukoliko je task uspešno kreiran vraća se pdPASS dok se u suprotnom vraća pdFALSE
  - Prilikom realizacije softvera obavezno proveravati šta vraća ova funkcija

Kratak opis ostalih bitnih API funkcija

- Brisanje kreiranog taska se vrši koristeći `vTaskDelete` funkciju koja ima sledeću deklaraciju

```
void vTaskDelete( TaskHandle_t pxTask );
```

Naziv parametra	Opis
pxTask	Instanca taska koji brišemo. Ukoliko prosledimo NULL parametar to podrazumeva da f-ju pozivamo iz taska koji želimo da brišemo

- Blokiranje taska na određeni period se realizuje koristeći `vTaskDelay` funkciju

```
void vTaskDelay( TickType_t xTicksToDelay );
```

Naziv parametra	Opis
xTicksToDelay	Broj **tikova** (poziva prekidnih rutina sistemskog tajmera) koliko želimo da task bude blokiran

Primer

➢ Zadatak 3

Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

Sa periodom od 100 tikova menja se stanje diode LD4, dok se sa periodom od 200 tikova menja stanje diode LD3 isključivo ukoliko je pritisnut taster S3.

Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena, ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

➢ Zadatak 3 - Rešenje

- Da bi došli do krajnjeg rešenja ovog zadatka potrebno je proći dve faze (koje zapravo oslikavaju realno rešavanje bilo kog praktičnog projekta u oblasti razvoja softvera za embedded aplikacije)
  1. **Faza projektovanja softvera (Teža, zahtevnija, iziskuje mnogo više vremena i znanja u odnosu na Fazu 2)**
     - ***(Podeli posao na taskove)*** - U okviru ove faze treba **detaljnije sagledati** zadatak kako bi se potencijalno izvršila **dekompozicija** zadatka na što jednostavnije logičke celine (koje odgovaraju nezavisnim taskovima)
     - **(Opiši funkcionalnost taskova)** - Kako bi nekome jasnije predstavili ideju usvojene realizacije treba koristiti dijagrame toka koji opisuju algoritme usvojene realizacije određenih logičkih celina (taskova)
     - **(Predstavi aktivnosti taskova u vremenu)** U cilju boljeg razumevanja ponašanja sistema za usvojene funkcionalnosti taskova, nacrtati dijagrame koji jasno pokazuju gde se koji task blokira, na čemu se blokira, kada task dobija procesorsko vreme, itd ...

Pimer

➢ Zadatak 3

Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

Sa periodom od 100 tikova menja se stanje diode LD4 dok se sa periodom od 200 tikova menja stanje diode LD3 isključivo ukoliko je pritisnut taster S3.

Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena, ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

➢ Zadatak 3 - Rešenje

- Da bi došli do krajnjeg rešenja ovog zadatka potrebno je proći dve faze (koje zapravo oslikavaju realno rešavanje bilo kog praktičnog projekta u oblasti razvoja softvera za embedded aplikacije)

  2. Faza realizacije softvera

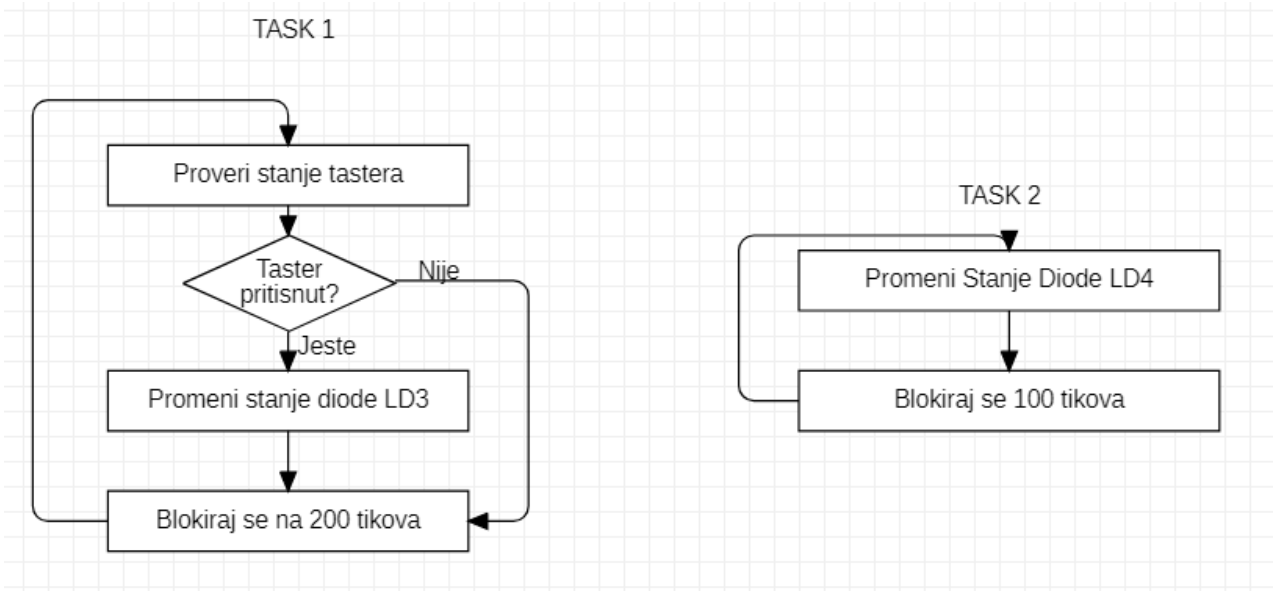
  - Ukoliko je 1. faza dobro realizovana onda u okviru ove faze ostaje posao „Kodovanja“ i verifikovanja rešenja.

- Jako je bitno razumeti posao koji treba da se uradi u okviru svake od faza jer to zapravo predstavlja i način na koji ćete rešavati i projektni zadatak.

Pimer

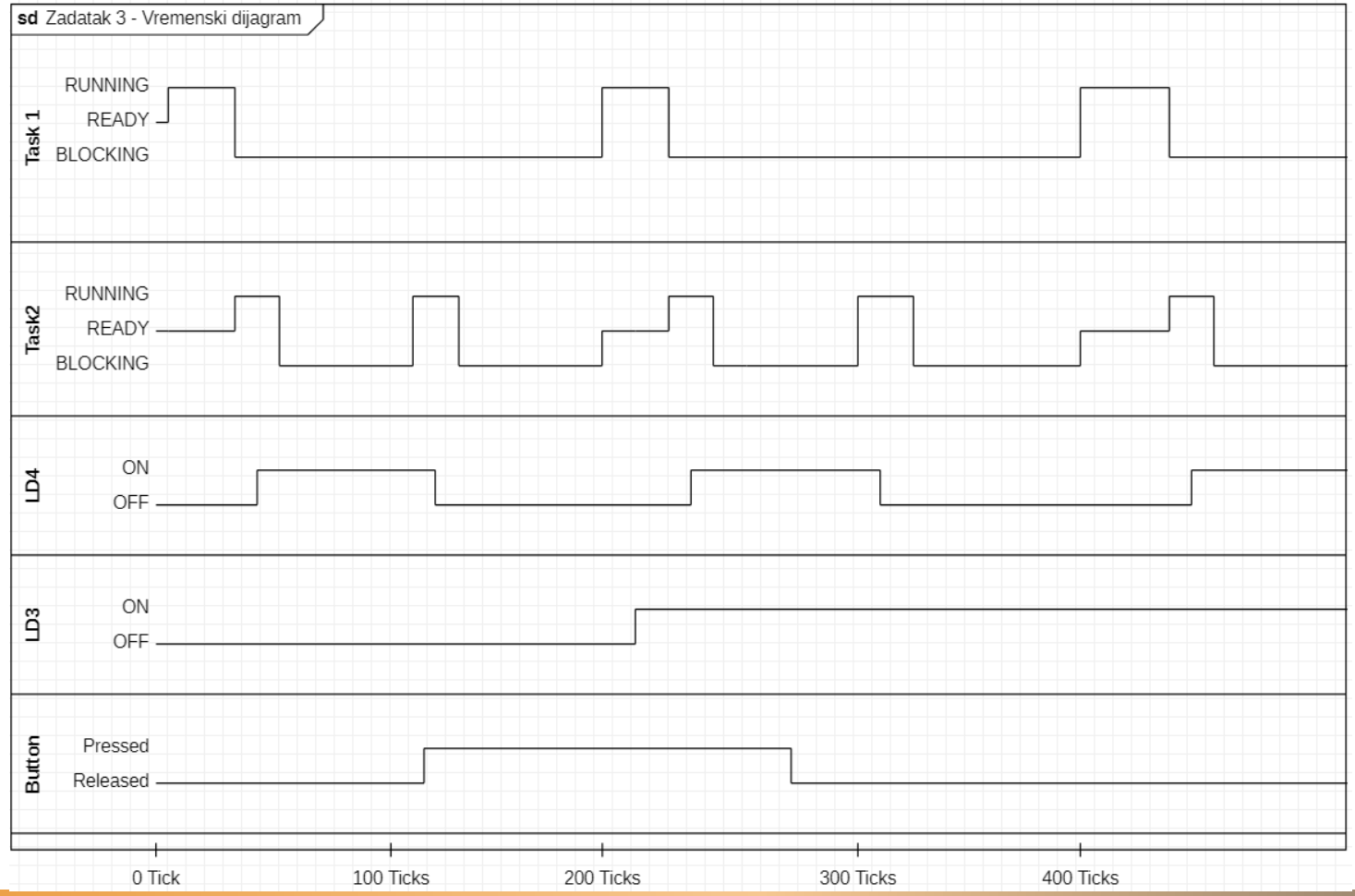
➢ Zadatak 3 – Rešenje (Projektovanje)

- U zadatku možemo uočiti dve logičke celine
  1. **Promena stanja diode na 200 sistemskih tikova i provera stanja tastera**
     - Implementiraćemo funkcionalnosti u okviru taska 1
  2. **Promena stanja diode na 100 sistemskih tikova**
     - Implementiraćemo funkcionalnosti u okviru taska 2
- Dijagram sekvence
  - U okviru ovog dijagrama opisan je implementirani algoritam



Pimer

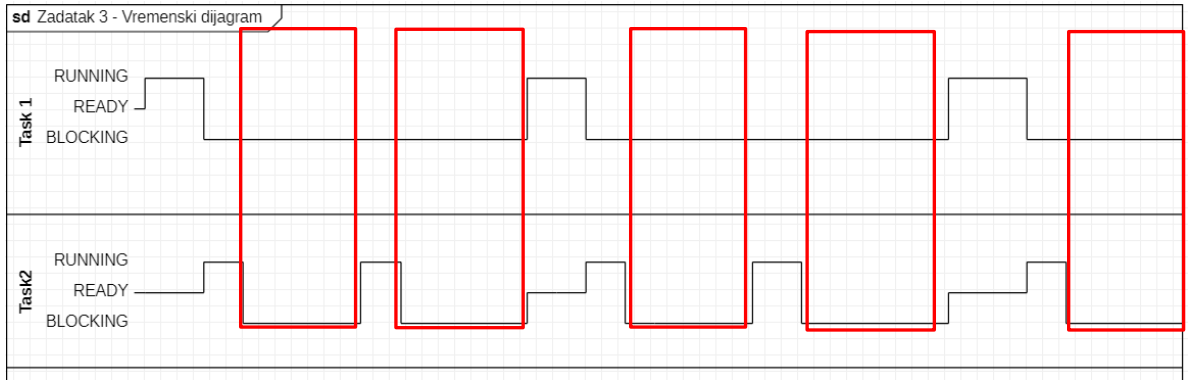
- Zadatak 3 – Rešenje (Projektovanje)
  - Dijagram koji opisuje ponašanje sistema u vremenu



Pimer

➢ Zadatak 3 – Rešenje (Realizacija)

- ...
- Pogledati ***SRV_2_1*** u okviru materijala
- Kako odrediti koliko 100 sistemskih tikova iznosi u ms?
- Šta se dešava u slučaju sistema baziranog na FreeRTOS-u ukoliko ne postoji nijedan task koji se izvršava?



- Od strane *FreeRTOS Sheduler*-a kreira se **IDLE** task da bi na sistemu postojao makar jedan task koji može da se izvršava.

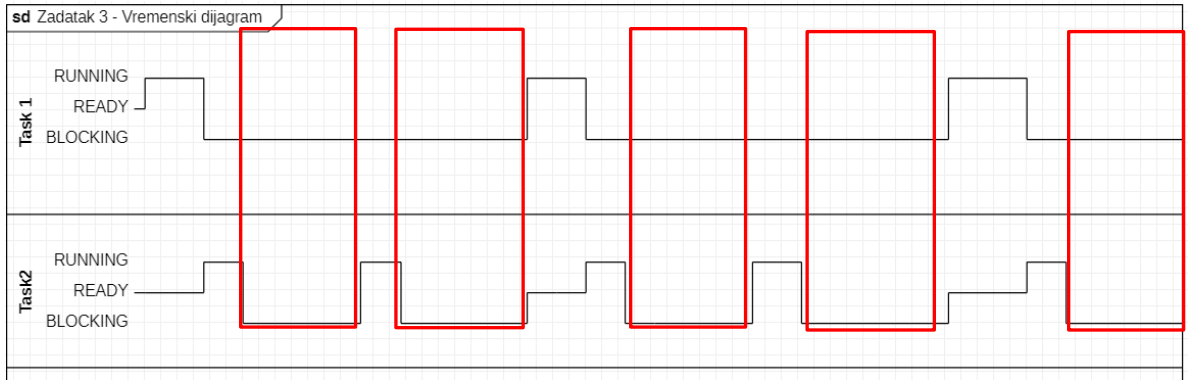
- IDLE task je task najnižeg prioriteta kako ne bi uticao na izvršaanje ostalih taskova

- Setovanjem makroa configUSE_IDLE_HOOK na 1 moguće je definisati callback funkciju koja će se pozivati iz IDLE taska. Ta callback funkcija mora imati sledeći potpis.

```
void vApplicationIdleHook( void );
```

Pimer

- Zadatak 3 – Rešenje (Realizacija)
  - ...
  - Pogledati ***SRV_2_1*** u okviru materijala
  - Kako odrediti koliko 100 sistemskih tikova iznosi u ms?
  - Šta se dešava u slučaju sistema baziranog na FreeRTOS-u ukoliko ne postoji nijedan task koji se izvršava?



- Kada se ova funkcija koristi za potrebe aplikacije, u okviru nje se **NE SME** koristiti nijedna API funkcija ili mehanizam koji bi izazvao blokiranje **IDLE taska koji zapravo poziva ovu funkciju**

Pimer

- **Zadatak 4**(Za samostalni rad)

  Vremenskim dijagramom opisati rad sistema ukoliko se zadatak 3 realizuje korišćenjem taskova istog prioriteta. Realizovati takvo rešenje.

- **Zadatak 5**(Za samostalni rad)

  Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

  Ukoliko se 3 puta pritisne taster S3 dioda LD3 menja stanje. Dioda LD4 sa periodom 100 sistemskih tikova menja stanje.

  Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena, ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

Pimer

➢ Zadatak 6

Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

Vrši ispis promenljive *data* na dvocifreni sedmosegmentni displej. Vrednost promeljive *data* se menja od strane korisnika iz CCS IDE alata. Pretpostaviti da korisnik nikada neće uneti vrednost veću od 100 tako da dodatne promene nisu neophodne.

Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena, ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

➢ Zadatak 6 - Rešenje

- Videti kod rešenja projekta ***SRV_2_2***

RTOS

KOMUNIKACIJA I SIGNALIZACIJA

SIGNALIZACIJA DOGAĐAJA KORIŠĆENJEM SEMAFORA

KORIŠĆENJE SEMAFORA U OKVIRU FREERTOSA

KOMUNIKACIJA I SIGNALIZACIJA DOGAĐAJA

- Do sada smo govorili o tasku kao nezavisnoj celini
- U većini slučajeva (skoro uvek) postoji potreba da se tasku **signalizira** pojava nekog događaja u sistemu (na primer iz prekidne rutine) ili da se da se izvrši **komunikacija** sa taskom u cilju razmene informacija.
- Za signalizaciju događaja u sistemu koriste se neki od sledećih mehanizama
  - **Semafor**
    - Najčešće binarni i brojački
  - **Notifikacija**
  - **Grupa događaja**
- Za komunikaciju sa taskom koriste se neki od sledećih mehanizama
  - **Mehanizam deljene memorije**
    - Najčešće u kombinaciji sa **mutex** semaforom
  - **Mehanizam prosleđivanja poruka**
    - ***Queue***

SIGNALIZACIJA DOGAĐAJA

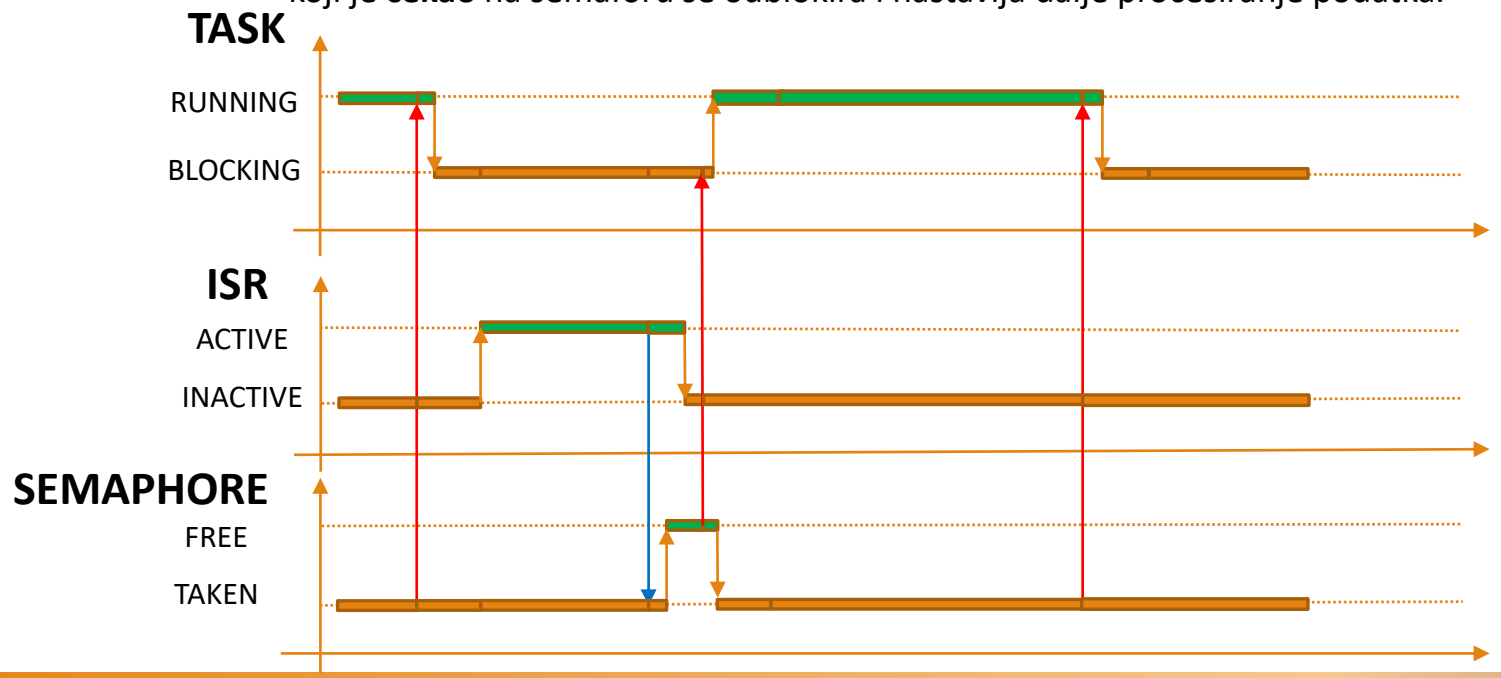
Mehanizam semafora

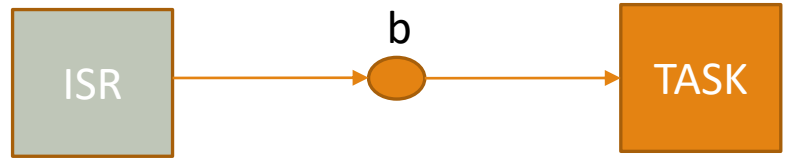
- Semafori se koriste kako bi se signalizirala pojava nekog događaja
  - Ovde isključujemo **mutex** tip semafora o kome će biti reči kasnije
- Neki od tipičnih primera korišćenja semafora su:
  - Signaliziranje događaja
    - Unutar prekidne rutine je završena preliminarna obrada podataka. Obradu podatka treba nastaviti u okviru taska. Na kraju prekidne rutine se **oslobađa** semafor i **task** koji je **čekao** na semaforu se odblokira i nastavlja dalje procesiranje podatka.
  - Sinhronizacija taskova
    - Task A je došao do dela algoritma gde treba da sačeka da se neki drugi task (TASK B) završi neku funkcionalnost.
  - Kontrola broja pristupa određenom resursu
    - Na primer, na nekoj embedded platformi koja koristi ethernet za komunikaciju, u slučaju TCP/IP steka može postojati ograničen broj trenutno otvorenih *socket* konekcija
- Dva tipa semafora koja se koriste za signalizaciju su
  - Binarni semafor
  - Brojački semafor

SIGNALIZACIJA DOGAĐAJA

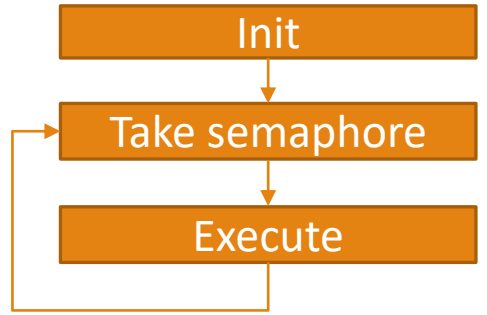
Mehanizam semafora

- Semafori se koriste kako bi se signalizirala pojava nekog događaja
  - Ove isključujemo **mutex** tip semafora o kome će biti reči kasnije
- Neki od tipičnih primera korišćenja semafora su:
  - Signaliziranje događaja
    - Unutar prekidne rutine je završena preliminarna obrada podataka. Obradu podatka treba nastaviti u okviru taska. Na kraju prekidne rutine se **oslobađa** semafor i **task** koji je **čekao** na semaforu se odblokira i nastavlja dalje procesiranje podatka.





Tok izvršavanja TASKA



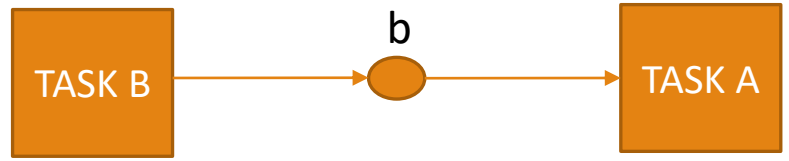
Na vremenskom dijagramu uočiti gde je došlo do signaliziranja tasku da se desio događaj?

SIGNALIZACIJA DOGAĐAJA

Mehanizam semafora

- Semafori se koriste kako bi se signalizirala pojava nekog događaja
  - Ovde isključujemo **mutex** tip semafora o kome će biti reči kasnije
- Neki od tipičnih primera korišćenja semafora su:
  - Sinhronizacija taskova
    - Task A čeka da se završi task B kako bi TASK A nastavio sa izvršavanjem
    - Vremenski dijagram je vrlo sličan prethodnom vremenskom dijagramu

    Za domaći nacrtati vremenski dijagram ponašanja sistema u okviru koga se jasno vidi sinhronizacija dva taska koristeći mehanizam semafora



  - Ograničavanje broja pristupa nekom resursu
    - U slučaju rada sa soketima u okviru TCP/IP steka često postoji ograničen broj trenutno otvorenih paralelnih konekcija.
- Za potrebe signalizacije događaja koriste se dva tipa semafora:
  - Binarni
    - Najčešće se koriste **za sinhronizaciju** i **signaliziranje** događaja
    - Predstavljaju **poseban tip brojačkih semafora**
  - Brojački
    - Najčešće se koriste za **pristup resursu koji ima ograničen broj paralelnih pristupa**
    - Nakon kreiranja, dodeljuje mu se **inicijalna vrednost brojanja** (svako uzimanje semafora smanjuje inicijalnu vrednost)

FreeRTOS

SEMAFORI

BINARNI I BROJAČKI SEMAFORI U FREERTOS-U

API FUNKCIJE ZA RAD SA OVIM SEMAFORIMA

PRIMERI

API Funkcije za rad sa semaforima

- U okviru *template* projekta potrebno je uključiti *semphr.h* header fajl ukoliko želimo da koristimo API funkcije FreeRTOS-a za rad sa semaforima
- API F-je za rad sa semaforima
  - Kao i u slučaju API funkcija za rad sa taskovima, navešćemo samo najčešće funkcije
  - U slučaju potrebe za korišćenjem nekih dodatnih API funkcija za rad sa semaforima, inkremnetalno ćemo ih uvoditi kroz prezentacije

Funkcija	Opis
`xSemaphoreCreateBinary`	*Kreira binarni semafor*
`xSemaphoreCreateCounting`	*Kreira brojački semafor*
`xSemaphoreTake`	*Zauzima semafor*
`xSemaphoreGive`	*Oslobađa semafor*
`vSemaphoreDelete`	*Briše semafor*

API Funkcije za kreiranje semafora

- Kreiranje binarnog semafora
  - Semafor se kreira pozivajući API funkciju FreeRTOS-a koja ima sledeću deklaraciju

```
SemaphoreHandle_t xSemaphoreCreateBinary( void );
```

  - Semafor **je kreiran kao „prazan“** što znači da prvo mora da se oslobodi (koristeći API funkciju `xSemaphoreGive`) pre nego što ga neko zauzme (koristeći API funkciju `xSemaphoreTake`)
  - Ukoliko je semafor uspešno kreiran vraća se instanca ***SemaphoreHandle_t*** strukture dok se u suprotnom vraća ***NULL***
- Kreiranje brojačkog semafora
  - Brojački semafor se kreira pozivajući API funkciju FreeRTOS-a koja ima sledeću deklaraciju

```
SemaphoreHandle_t xSemaphoreCreateCounting(
                  UBaseType_t uxMaxCount,
                  UBaseType_t uxInitialCount );
```

Naziv parametra	Opis
`uxMaxCount`	Maksimalan broj do kojeg semafor može da broji. Kada se dostigne ova vrednost, semafor više ne može da se oslobađa.
`uxInitialCount`	Početna vrednost brojača

  - Ukoliko je semafor uspešno kreiran vraća se instanca ***SemaphoreHandle_t*** strukture dok se u suprotnom vraća ***NULL***

API Funkcije za zauzimaje i oslobađanje semafora

- Iste funkcije se koriste bez obzira da li je u pitanju binarni ili brojački semafor
  - Semafor se **zauzima** pozivajući API funkciju FreeRTOS-a koja ima sledeću deklaraciju

```
BaseType_t xSemaphoreTake( SemaphoreHandle_t xSemaphore,
                           TickType_t xTicksToWait );
```

Naziv parametra	Opis
xSemaphore	Instanca prethodno kreiranog semafora koji se zauzima
xTicksToWait	Maksimalan broj tikova koliko task može da se blokira čekajući da semafor postane dostupan

  - Funkcija vraća pdPASS ako je semafor uspešno zauzet a pdFAIL ukoliko zauzimanje semafora nije uspešno urađeno
  - Funkcija se poziva isključivo iz taska
  - Semafor se **oslobađa** pozivajući API funkciju FreeRTOS-a koja ima sledeću deklaraciju

```
BaseType_t xSemaphoreGive( SemaphoreHandle_t xSemaphore );
```

Naziv parametra	Opis
xSemaphore	Instanca prethodno kreiranog semafora koji se oslobađa

  - Funkcija vraća pdPASS ako je semafor uspešno oslobođen, a pdFAIL ukoliko oslobađanje semafora nije uspešno urađeno

➢ ZADATAK 7

Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

Pritisak tastera S3 uzrokuje promenu stanja diode LD3.

Zadatak je potrebno realizovati korišćenjem barem jednog semafora.

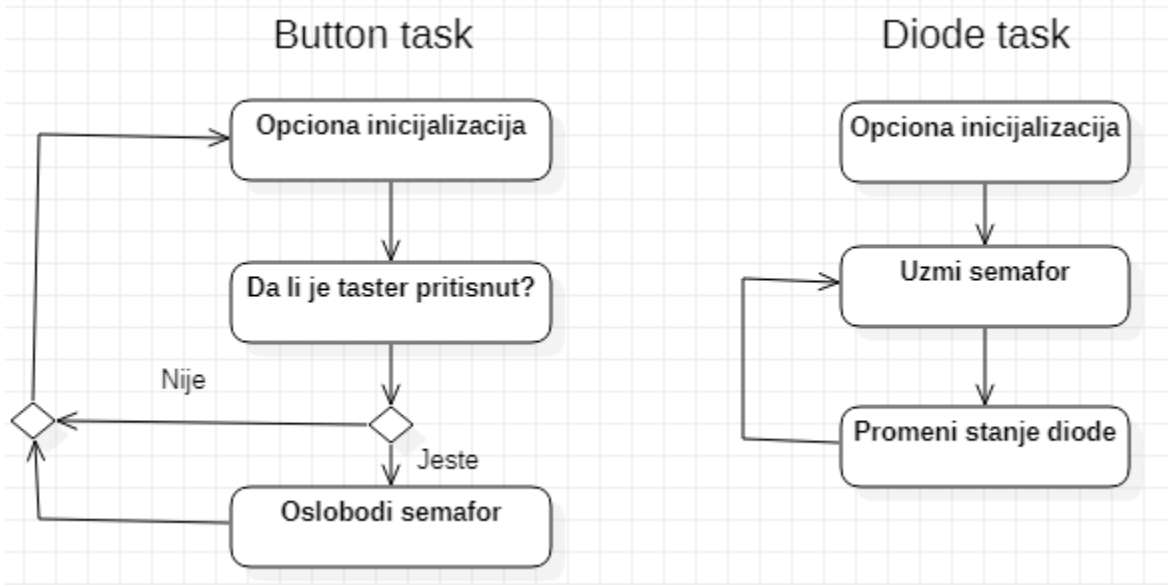
Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena, ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

➢ Zadatak 7 – Rešenje (Projektovanje)

- U okviru zadatka moguće je uočiti dve logičke celine:
  1. **Detekcija pritiska tastera**
     - Ovu logičku celinu ćemo implementirati u okviru taska „Button task“
  2. **Promena stanja diode**
     - Ovu logičku celinu ćemo implementirati u okviru „LE Diode task“
- Dakle, zadatak „Button task“ taska jeste da detektuje pritisak tastera i da informaciju o događaju „pritisak tastera“ prosledi tasku „LE Diode task“ koji ima zadatak da menja stanje diode. Signaliziranje događaja „taster pritisnut“ realizovaćemo **koristeći binarni semafor**.

➢ Zadatak 7 – Rešenje (Projektovanje)

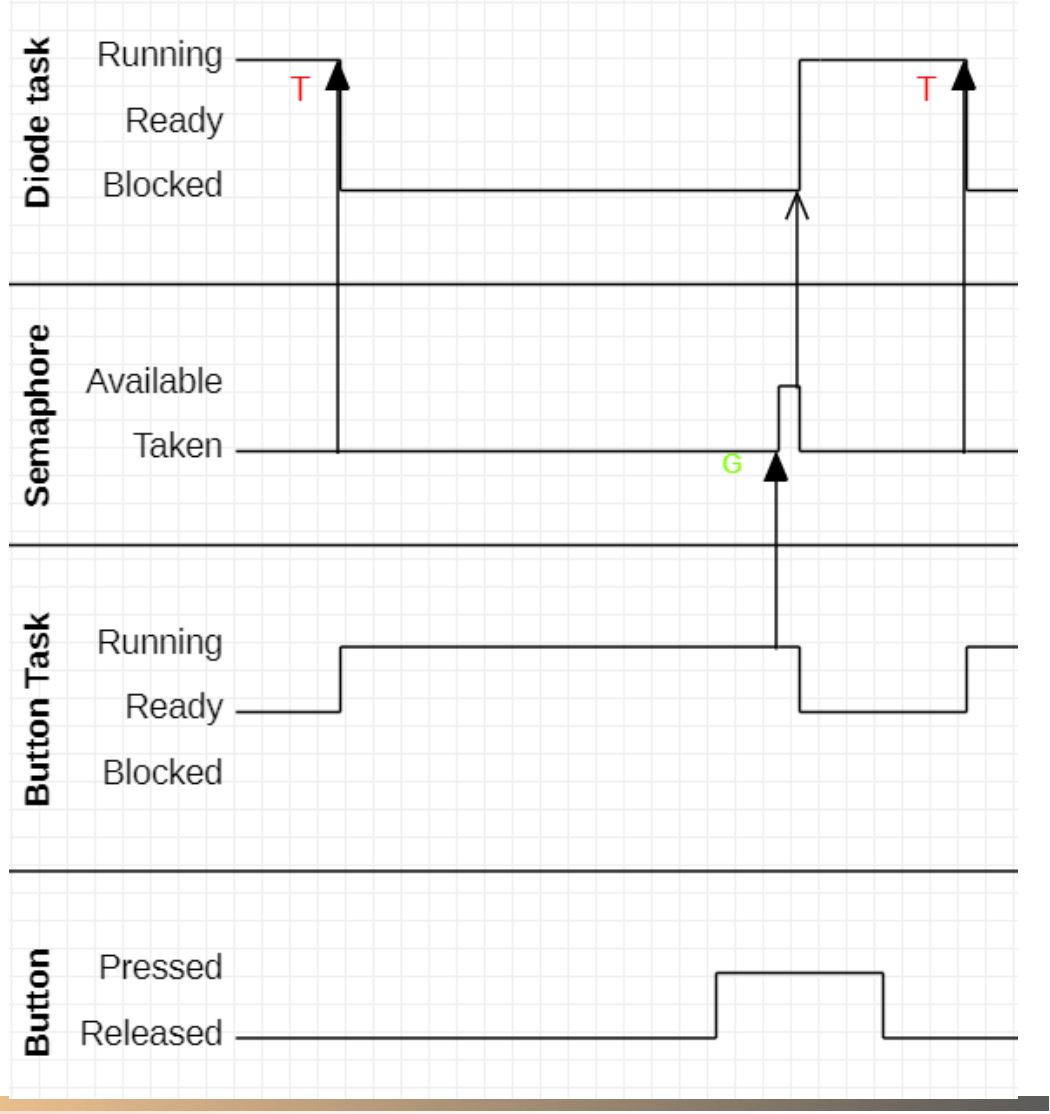
- Pre nego krenemo u realizaciju potrebno je opisati algoritme koristeći **dijagram sekvence,** a zatim opisati izvršavanje softvera u vremenu koristeći **vremenske dijagrame**.
- **Dijagrami sekvence** koji opisuju algoritame u okviru svakog od taskova:



- **Kom tasku dodeliti veći prioritet?**

PRIMERI

- Zadatak 7 – Rešenje (Projektovanje)
  - Vremenski dijagram koji opisuje ponašanje sistema za slučaj kada je prioritet „Diode task“ taska veći od prioriteta „Button task“ taska
  - **Kako bi izgledao vremenski dijagram u slučaju da task „Diode Task“ ima manji prioritet od taska „Button task“?**
    - Nacrtati vremenski dijagram za ovaj slučaj



- **G** – Task oslobađa semafor
- **G*i*** – Task oslobađa semafor ***i***-ti put
- **T** – Task uzima semafor

- Zadatak 7 – Rešenje (Realizacija)
  - Pogledati kod ***SRV_2_3*** projekta

- **ZADATAK 8**

Projektovati i **realizovati** softver baziran na *FreeRTOS*-u koji treba da zadovolji sledeću funkcionalnost:

Pritisak tastera S3 uzrokuje **inkrementiranje brojačke promenljive** koja broji po modulu 10. Vrednost brojačke promenljive se **ispisuje na jednom 7seg** LE dipsleju. Dioda LD3 menja stanje nakon što brojačka promenljiva izbroji do broja 9.

Zadatak je potrebno realizovati korišćenjem **barem** jednog semafora.

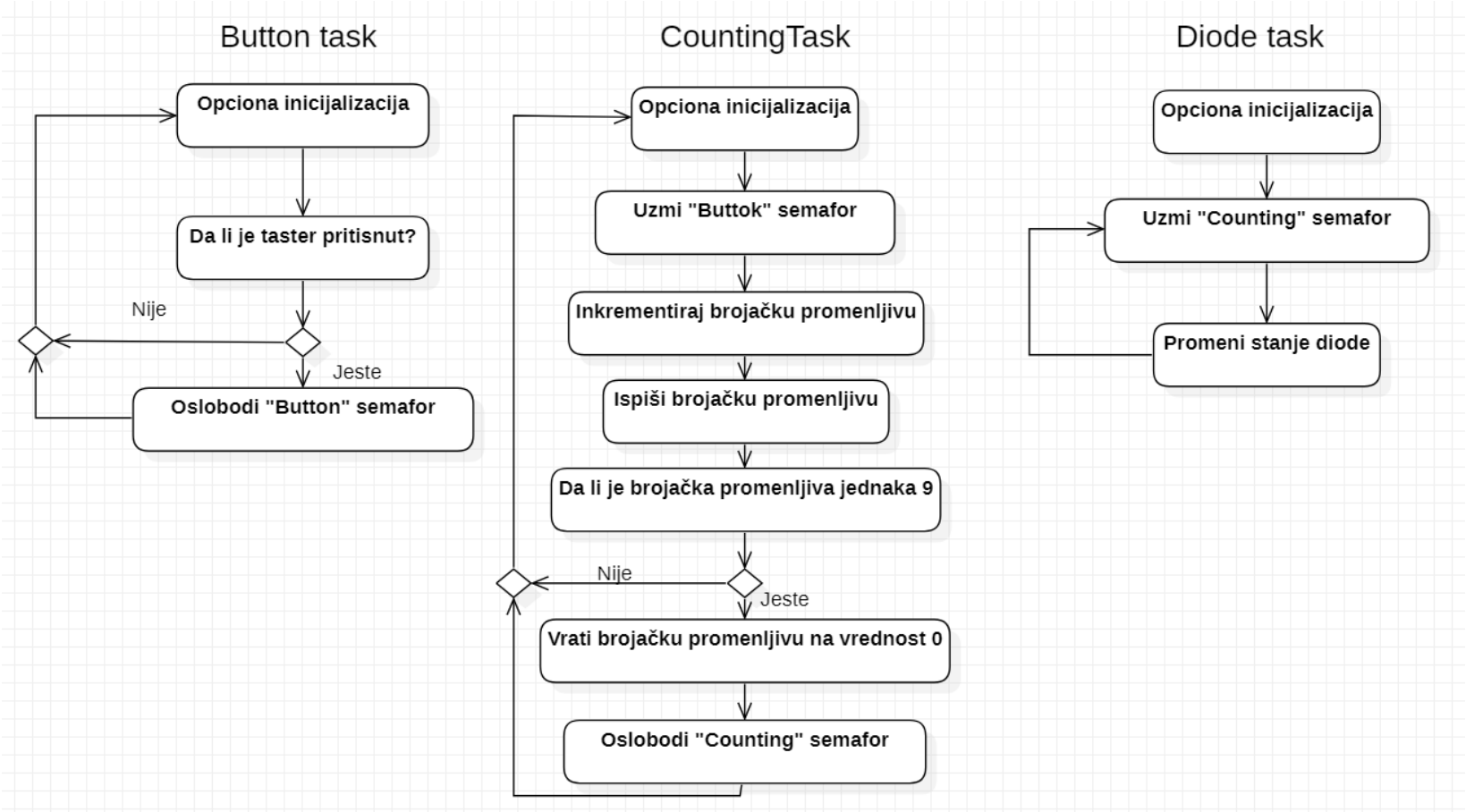
Za potpuno realizovano rešenje potrebno je priložiti odgovarajuće dijagrame koji što tačnije opisuju ponašanje sistema tokom vremena ali i implementirane algoritme. Pored toga, treba priložiti i kod realizovanog rešenja.

➢ Zadatak 8 – Rešenje (Projektovanje)

- U okviru zadatka moguće je uočiti tri logičke celine:
  1. **Detekcija pritiska tastera**
     - Ovu logičku celinu ćemo implementirati u okviru taska „Button task“
  2. **Kontrola brojačke promenljive i ispis na 7seg LE displej**
     - Ovu logičku celinu ćemo implementirati u okviru taska „Counting task“
  3. **Promena stanja diode**
     - Ovu logičku celinu ćemo implementirati u okviru „LE Diode task“
- Dakle, zadatak „Button task“ taska jeste da detektuje pritisak tastera i da informaciju o događaju „pritisak tastera“ prosledi tasku „Counting task“. Signaliziranje događaja „taster pritisnut“ realizovaćemo **koristeći binarni semafor**. „Counting Task“ ima zadatak da inkrementira brojačku promenljivu, ispisuje sadržaj brojačke promenljive na 7seg LE displej i da detektuje kada vrednost brojačke promenljive dostigne broj 9. Informacija da je brojačka promenljiva dostigla vrednost 9 se prosleđuje „LE Diode Task“ tasku kroz drugi **binarni semafor** na koji je ovaj task blokiran.

PRIMERI

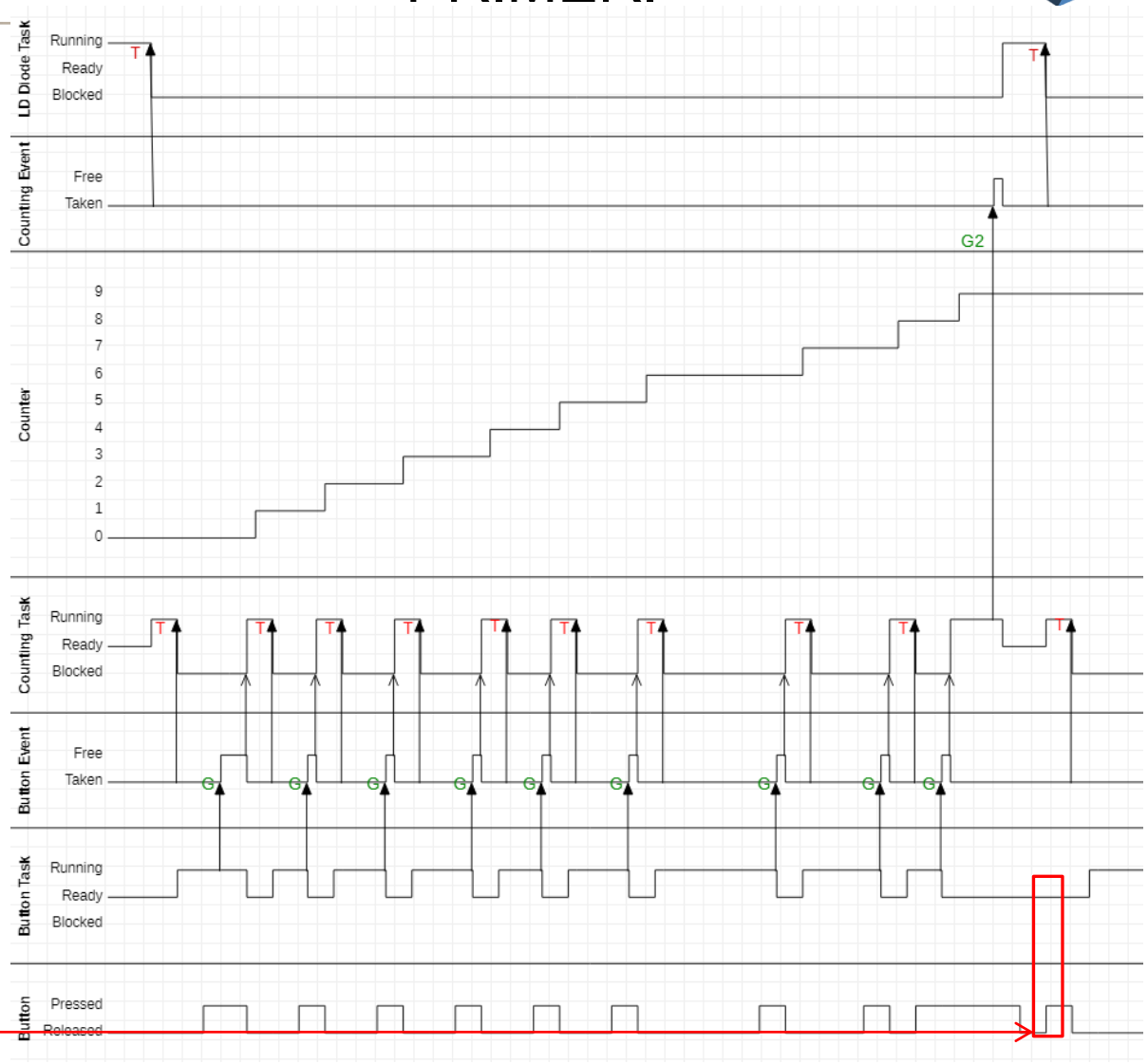
- Zadatak 8 – Rešenje (Projektovanje)



FreeRTOS SEMAFORI

PRIMERI

- Zadatak 8 – Rešenje (Projektovanje)
  - **G** – Task oslobađa semafor
  - **G*i*** – Task oslobađa semafor ***i***-ti put
  - **T** – Task uzima semafor



Potencijalni problem

- Sada se jasno vidi da je detaljna faza projektovanja jedan od načina da razumemo **potencijalne probleme u dizajnu** softvera

- Zadatak 8 – Rešenje (Realizacija)
  - Videti primer ***SRV_2_4***

FreeRTOS

KORISĆENJE PREKIDNIH RUTINA

KAKO SE KORISTE PREKIDI ZAJEDNO SA FREERTOSOM

ZAMENA KONTEKSTA U OKVIRU PREKIDNE RUTINE

PRIMERI