Projektni zadatak

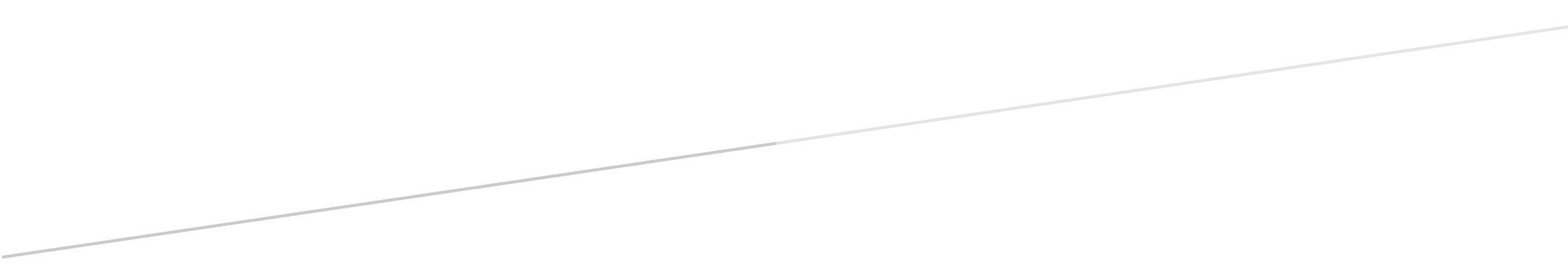
Rosless jog

Aleksa Đurđević

Mihailo Dikanović

Jelena Gajić

Luka Misojčić



**Univerzitet u Novom Sadu Januar 21,2024**

**Fakultet Tehničkih Nauka**

**Novi Sad**

1. **Uvod**

* Problem koji je trebalo resiti

U svrhu ovog projekta koristili smo kod sa postojećih vezbi 7.U ovom projektu razvijen je upravljač za manipulaciju servo motorima u zglobovima robotske ruke, bez upotrebe ROS (Robot Operating System) okruženja koji sluzi za komunikaciju između korisnika i računarskih resursa.

Upravljanje servo motorima ostvareno je putem PWM digitalnog signala sa periodom od 20ms. Duty Cycle, odnosno faktor ispunjenosti signala, koristi se za podešavanje ugla servo motora.U specifičnoj konfiguraciji projekta, raspon faktora ispunjenosti [10%, 90%], ili u promilima [100‰, 900‰], mapira se na ugao servo motora u opsegu [-90°, 90°], odnosno [-π/2, π/2]. Ovaj opseg faktora ispunjenosti naziva se "široki faktor ispunjenosti", koji servo motor direktno ne tumači.

Kroz upravljački program, korisnički proces postavlja željeni faktor ispunjenosti putem odgovarajuće datoteke, dok upravljač generiše PWM signal na određenim pinovima. Taj signal se dalje šalje Servo Motor Shield-u, koji ga prevodi u faktor ispunjenosti [25‰, 125‰], razumljiv servo motoru.

* Zadatak naseg projekta:
  + - Citanje inputa sa joystick-a
    - Implementirati funkcije za upravljanje servo motorima korišćenjem analognog inputa sa joystick-a.
    - Razvijemo sistem za upravljanje servo motorima u robotskoj ruci, omogućavajući precizno pozicioniranje i kontrolu svakog zgloba.

1. **Joystick**

* **Testiranje joystick-a**

Za pocetak, bilo je potrebno testirati sam joystick i vrednosti koje on zadaje.

To smo uradili pomocu testne aplikacije i zabeležili smo sledece vrednosti:

1. Za selekciju:
   * + Shuffle Up - Button 4(LB)
     + Shuffle Down - Button 6 (LT)
     + Menjanje brzine okretanja – Button 4 i Button 2

2) Za pokretanje:

* + - Axis 1 - Gore
    - Dole

Korišćena je datoteka koja upravlja joystickom , libenjoy. Kroz sam kod program se inicajlizuje, ažurira se lista joystick-a,otvara prvi joystick u listi i zatim konstantno proverava događaje poput pomeranja ose ili pritiska dugmeta na kontroleru.

1. **Korišćene biblioteke i datoteke**

Za kontrolu inputa sa joysticka koristimo biblioteku 'libenjoy.h'(<https://github.com/Tasssadar/libenjoy>). U okviru datoteka libenjoy.h i libenjoy.c najbitnije funkcije koje smo implementriali u okviru same logike koda bile su:

* libenjoy\_enumerate: Popunjava listu info\_list sa informacijama o trenutno povezanim džojsticima.
* libenjoy\_poll: Čeka na događaje sa džojstika i vraća ih kao libenjoy\_event. Može takođe popuniti buffere događajima.

Iz standradne biblioteke iskoristili smo funkcije kao sto su:

* lseek: Postavlja poziciju trenutnog mesta unutar fajla gde će sledeća operacija čitanja ili pisanja biti izvršena.
* Read: Čita podatke iz fajl deskriptora
* Write: Piše podatke u fajl deskriptor

1. **Logika koda**

* **Rešenje problema**

Sama logika koda implementirana je u ’rossles\_jog.c’. U okviru koda koristimo promenljivu selectedServo koja nam pokazuje koji motor je trenutno odabran i selectedSpeed koja nam omogućava da izaberemo brzinu. Pomoću joystick-a izaberemo koji motor želimo da koristimo i zatim pomeramo robotsku ruku u odredjenom pravcu za ugao koji je unapred definisan u okviru koda.Što se tiče 'wscript.py' nju je bilo potrebno izmenti u svrhu linkovanja datoteka radi pokretanja samog programa.

Koraci kako je realizovan kod:

1. **Inicijalizacija:**

* Inicijalizuje se libenjoy kontekst.
* Popunjava se lista povezanih džojstika.
* Otvara se fajl koji predstavlja drajver servo motora,ukoliko nije uspešno otvaranje ispiše se error.

1. **Postavljanje Početnih Vrednosti:**

* Postavili smo odredjene početne vrednosti za sam polozaj ruke(centar).
* Namestili smo tako da nam se pri svakom ponovnom pokretanju programa ruka vrati u početni položaj.

1. **Povezivanje sa Džojstikom:**

* Program pokušava uspostaviti komunikaciju sa prvim džojstikom iz liste.

1. **Glavna Petlja:**

* Čeka se na događaje sa džojstika.
* Ako se promeni položaj ose na džojstiku, ažuriraju se vrednosti duty ciklusa servo motora u skladu s pravcem promene.
* Ako je pritisnut taster, ažurira se trenutni servo motor ili trenutna brzina okretanja
* Prate se i promene statusa povezivanja džojstika.

1. **Ažuriranje Kontrole Servo Motora:**

* Duty ciklusi servo motora se ažuriraju na osnovu promena položaja ose ili pritiska tastera.
* Ažuriranja se vrše u skladu s definisanim vrednostima kao što su DELTA\_DUTY (implementirali smo je posebno, kao formulu za promenu ugla)i širina duty ciklusa.
* Definisane su maksimalna, minimalna i srednja wide\_servo vredsnoti duty-ja

1. **Oslobađanje Resursa:**

* Nakon završetka rada, oslobađaju se resursi džojstika i zatvara se fajl koji predstavlja drajver servo motora.

7.**Dodatne funkcionalnosti koje smo implementirali**:

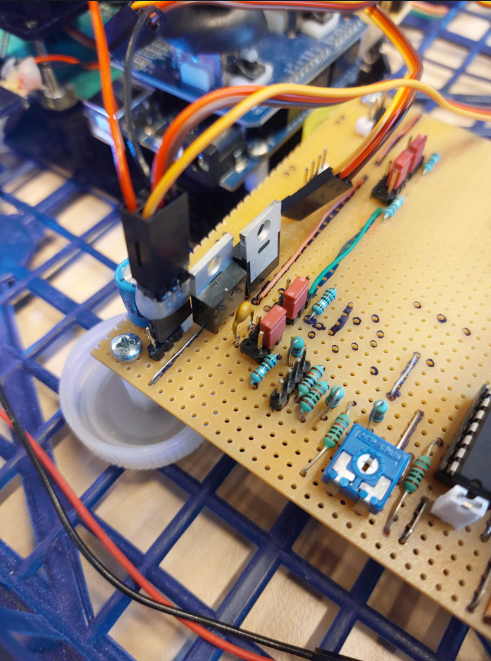
* Nalaze se u okviru datoteka ’config.h’ , ’tools.h’ i ’tools.c’
* Neke od funkcija koje smo implementirali u tools.c jesu  *WideToServoDuty* koja sluzi da konvertuje vrednosti iz wide\_duty u servo\_duty.Pored toga napravili smo i funkciju DebugInfo koji nam ispisuje poruke na standardni izlaz u slučaju da je uključen debug mode.
* Kod datoteke config.h implementirano je sledeće:
  + - WIDE\_DUTY -definiše da li se koristi wide duty ili sevo duty
    - SERVO\_SPEEDS - brzina okretanja serva
    - DEBUG\_MODE -da li treba ispisivati debug podatke (potvrda nakon uspesnog izvrsavanja blokova koda)
    - SELECTED\_JOYSTICK LEFT\_JOYSTICK -koji dzojstik upravlja rukom
    - SELECTED\_NEXT\_BUTTON LEFT\_BUMPER -koje dugme sluzi za šaltanje serva unapred
    - SELECTED\_PREVIOUS\_BUTTON LEFT\_TRIGGER -koje dugme sluzi za šaltanje serva unazad
    - REMEMBER\_PAST\_STATE - da li da ruka nastavi tamo gde je stala nakon gasenja i ponovnog pokretanja programa, ili da se vrati na pocetnu poziciju. Bitno je da 'motor\_ctrl' mora da radi sve vreme da bi ovo radilo.

1. **Pokretanje programa**

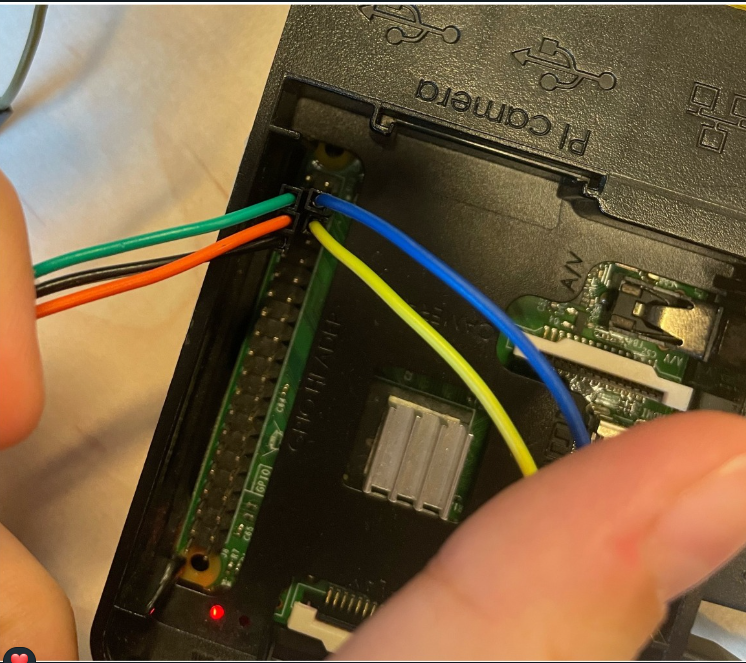
* **Kompajliranje**

Da bi porgram uspešno bio pokrenut, prvo je trebalo odraditi sva prethodna testiranja kao sto su gore navedena za sam joystick, potom pokretanje ’motor.ctrl’, za ovo su korišćeni make i make start , i na samom kraju su uradjeni ./waf configure i ./waf build radi pokretanja samog koda.

* Testiranje

Da bi proverili kako nas kod radi, i da li radi ispravno bilo je potrebno povezati RPi2 na Servo Motor Shield. 

Nožice odnosno pinove bilo je potrebno povezati na sledeći način:



Nakon uradjenih svih ovih koraka ruka je bila uspešno pokrenuta uz pomoć joystick-a.

1. Poteškoće pri izradi i zaključak

Poteškoće sa kojima smo se susreli pri samoj izradi projekta, za početak bila je ta što se sa ovakvom vrstom projekta do sad nismo susreli, bilo je potrebno pogledati sve unapred zadate fajlove, kako bismo mogli implementirati dalje logiku, istražiti svaku funkciju i njene parametre radi njihovog korišćenja. Takodje je postojalo dosta datoteka koje je trebalo koristiti sa kojima se ranije nismo susretali, na primer koriscenje waf-a za kompajliranje. Oko same logike koda nismo imali puno prepreka.