|  |  |  |
| --- | --- | --- |
|  | **UNIVERZITET U NOVOM SADU  FAKULTET TEHNIČKIH NAUKA** |  |

|  |
| --- |
| **Departman za računarstvo i automatiku**  **Smer računarstvo i automatika**  **Odsek za računarsku tehniku i računarske komunikacije**  **Control every motor from separate thread**  **Autori: Gorana Marinković RA161/2020**  **Nikolina Mihaljovski RA35/2020**  **Mentor: Miloš Pilipović**  **Predmet: Operativni sistemi za rad u realnom vremenu** |
| **Novi Sad, januar 2023.** |

**Sadržaj**

[UVOD 3](#_Toc125063933)

[KONCEPT REŠENJA 4](#_Toc125063934)

[POKRETANJE MOTORA 7](#_Toc125063935)

[ZAKLJUČAK 9](#_Toc125063936)

# **UVOD**

Cilj ovog projektnog zadatka jeste kontrolisanje četiri servo motora pomoću različitih niti, upotrebom Raspberry Pi2 mikrokontrolera. Raspberry Pi na sebi ima GPIO pinove koji omogućavaju komunikaciju sa nekim periferijskim jedinicama, u našem slučaju servo motorom.

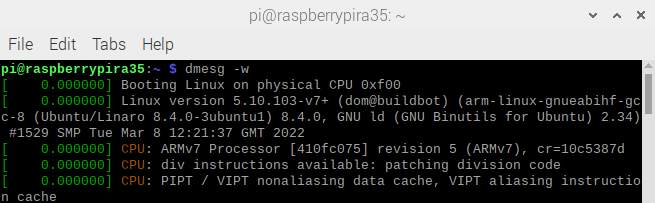
# **KONCEPT REŠENJA**

Izradu projektnog zadatka smo započele analizom sadržaja vežbe 5 i vežbe 7. Zbog same kompleksnosti i sadržajnosti datih fajlova, odlučile smo da iste optimizujemo i napravimo nove .c i .h fajlove u kojima smo implementirali rešenje.

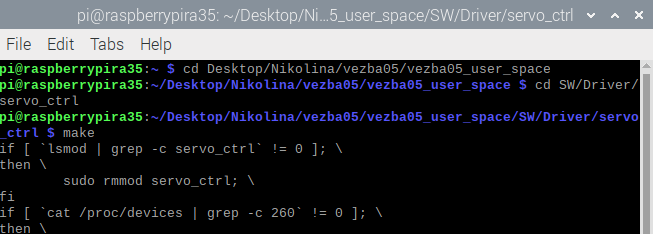
Na samom početku bilo je neophodno naše datoteke povezati sa hardverom u računaru, za šta smo koristile driver dat u vežbi 5 (***SW/Driver/servo\_ctrl***). On omogućava operativnom sistemu da komunicira sa servo motorima i da kontroliše njihove pokrete. Bez ovog driver-a operativni sistem ne bi bio u stanju da razume i kontroliše servo motore. To smo uradile na sledeći način:

***#define SERVO\_DRIVER “/dev/servo\_ctrl”***

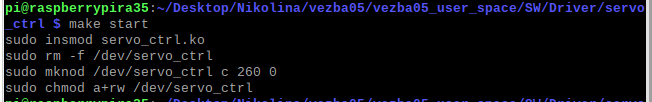
Da bi se povezivanje uspešno izvršilo, potrebno je prvo pokrenuti driver. Na sledećim slikama (*Slika1, Slika2, Slika3*) je prikazan postupak toga:



*Slika 1*



*Slika 2*



*Slika 3*

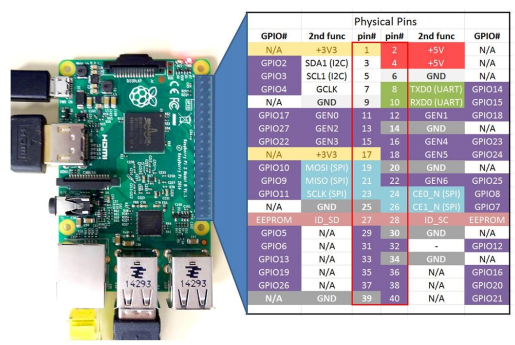
PWM (Pulse Width Modulation) signal je elektronski signal koji se koristi za kontrolisanje uređaja, ujedno i servo motora. On se sastoji od ciklusa koji se sastoji od dva dela: ***on*** i ***off*** perioda. “On“ period se naziva ***duty*** i predstavlja vremekoje signal ostaje u “on“ stanju tokom jednog ciklusa. Konkretno u našem slučaju ciklus smo postavile od 25 do 125 i izdelile ga na 20 koraka od kojih svaki predstavlja 5% ciklusa.

Servo motori kojima upravljamo u kodu su predstavljeni strukturom koja sadrži polja ***servoId*** i ***servoDuty.*** Menjanjem servoDuty-a mi zapravo kontrolišemo servo motore. Svaki novi korak povećava servoDuty za 5, a potom uspavljuje nit na 1 sekund, čime postižemo lakše praćenje kretanja servo motora.

Funkcionalnost niti se ogleda u tome što nakon 5 koraka prvog pokrenutog motora, pokreće se naredni motor i to važi za sva 4 servo motora.

# **POKRETANJE MOTORA**

Na osnovu date šeme (*Slika4*) prvenstveno je bilo neophodno preko pinova povezati servo motor sa Raspberry Pi2.

**

*Slika 4*

Nakon povezivanja servo motora na računar, potrebno je pokrenuti naš zadatak (sa prethodno pokrenutim driver-om) (*Slika5*).



*Slika 5*

Na osnovu prikazanog ispisa u terminalu, zaključujemo da je kontrolisanje servo motora, preko niti, uspešno realizovano u ovom projektu. Nakon čega možemo prekinuti izvršavanje programa.

# **ZAKLJUČAK**

Finalni rezultat projekta u potpunosti ispunjava zahteve koji su od nas traženi. Dobijena je veoma skladna “ koreografija“ servo motora.

Tokom izrade ovog projekta, većinski fokus je bio na konkurentnom programiranju u programskom jeziku C, ali smo i pored toga dodatno razumele i shvatile na koji način se PWM signalima upravlja servo motorima. Takođe, razumele smo komunikaciju između Raspberry Pi-a i periferijskih uređaja (konkretno servo motora) sa kojima se spaja preko GPIO pinova kao i kako se istim upravlja.