Univerzitet u Beogradu
Elektrotehnički fakultet

Marko Ljubisavljević i Luka Orlandić, 2014/0620 i 2015/0390

Semaforizovana raskrsnica

projekat iz predmeta Principi modernih telekomunikacija

mentor:
prof. dr Milan Bjelica

Beograd, avgust 2018.

Sažetak

U okviru predmeta Principi modernih telekomunikacija (u daljem tekstu PMT) realizovali smo Semaforizovanu raskrsnicu koju je moguće konfigurisati iz Grafičkog korisničkog interfejsa (GUI-a). Projekat predstavlja direktnu spregu hardware-a (*kombinacionih prekidačnkih mreža*) i software-a (*programiranje i sinhronizacija raskrsnice u programskim jezicima C i Python*). Na kraju kao rezultat projekta, raskrsnica sa semaforima je sinhronizovana i spremna za korišćenje u realnom vremenu.

Ključne reči: Semafor, Raskrsnica, C, Python, UART

Sadržaj

Slike

Glava 1

Uvod

1.1 Cilj & zamisao projekta

Veoma je intuitivno odrediti cilj ovog projekta: Napraviti jednostavnu semaforizovanu raskrsnicu sa semaforima za automobile i pešake koji su međusobno sinhronizovani. Prilikom sinhronizacije potrebno je voditi računa o praktičnim problemima (npr. trepćuće zeleno koje se pali onda kada pravac kojim se kreću automobili treba da izgubi mogućnost prolaska) i mnogim drugim.

1.2 Hardware

Za paljenje svetala na semaforima koristili smo mikrokontroler TI **MSP430G2553**. Dodatne elektronske komponente koje smo korisitili u realizaciji su kombinacione prekidačke mreže SN74LS04 (*NOT gate*), SN74HCT139 (*Dual 2-input Decoder*) i SN74HCT32 (*OR gate*).

Slika 1.1: *Logička kola korišćena u izradi projekta.*

1.3 Software

Logičko ”povezivanje” semafora kao i paljenje svetala na njima koristi osnove jezika C i Python. Mikrokontroler je programiran u C-u (za njega je napravljen manji skup funkcija koji omogućava lakše korišćenje) a odlučivanje koje svetlo će na kom semaforu biti upaljeno - za to je odgovoran Python3.7. Komunikacija između mikrokontrolera i korisničkog računara se odvija putem *USCI UART komunikacije*.

Glava 2

Projektni zadatak

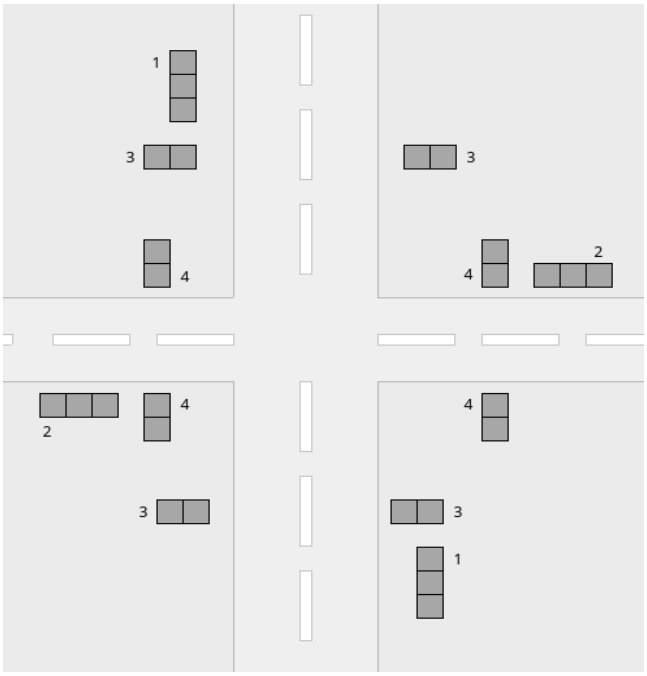
Kao što je već rečeno, potrebno je napraviti semaforizovanu raskrsnicu i osposobiti je da bude od praktičnog značaja. Rezultat izrade treba da bude jednostavna aplikacija koja omogućava korisniku upravljanje i naprednu sinhronizaciju na samoj raskrsnici.

2.1 Semaforizovana raskrsnica

Kao model raskrsnice koja je korišćena u ovom projektu uzeta je ona u kojoj se ukrštaju dva putna pravca po devedeset stepeni. Postoji ukupno 4 automobilska i 8 pešačkih semafora i to:

- 2 x 2 semafora za automobile po jednom putnom pravcu
- 2 x 4 semafora za pešake po jednom putnom pravcu

Semafori S1 i S2 su automobilski semafori koji prate odgovarajuće putne pravce. Pešački semafori koji su upareni uz njih su SP3 (*Semaphore Passengers*) i SP4. U nastavku je data slika raskrsnice koja je korišćena u projektu.



Slika 2.1: *Šema raskrsnice korišćena u projektu.*

2.2 Kombinaciona prekidačka mreža

Sve komponente korišćene u ovom projektu su proizvedene od strane *Texas Instruments*-a.

2.2.1 Semafor za automobile

Kako bi omogućili semaforu da svetli bilo je potrebno napraviti omanju prekidačku mrežu. U okviru jednog ciklusa rada semafora na njemu postoje 4 različite situacije, i to (predstavljeni binarnim vrednostima):

- 00 - Crveno + Žuto
- 01 - Zeleno
- 10 - Žuto
- 11 - Crveno

Situacija kada na semaforu trepće žuto ili zeleno obrađena je softverski o čemu će biti reči u nastavku.
Kako postoje 4 kombinacije vrednosti potrebne da se opiše stanje semafora,

dovoljan nam je bio dekoder ***SN74HCT139*** *Dual 2-input Decoder*. Izlazi dekodera daju invertovanu vrednost (*active low*), što nam dodatno povećava kombinacionu šemu. Kako bi rešili ovaj problem dodali smo invertor **SN74LS04**.
Potrebno je odrediti u kojim situacijama se pali određeno svetlo.
Crveno svetlo se pali kada je aktivno:

- 00 - Crveno + Žuto, **ili**
- 11 - Crveno

Žuto svetlo se pali kada je aktivno:

- 00 - Crveno + Žuto, **ili**
- 10 - Žuto

Gornji prikaz situacija kada se pojavljuje određeno svetlo govori da nam je potrebno jedno OR logičko kolo. U našem projektu smo koristili ***SN74HCT32*** *(Quad 2-input OR Gate)* kako bi povezali signalizaciju za žuto i crveno svetlo. Zeleno svetlo se pali samo u jednoj situaciji - kada je ulaz dekodera *0b01* - pa izlaz invertora možemo direktno da povežemo na zelenu LE diodu na šemi. **Enable bit** samog dekodera je aktivan kada je povezan na masu (*active low*).

2.2.2 Semafor za pešake

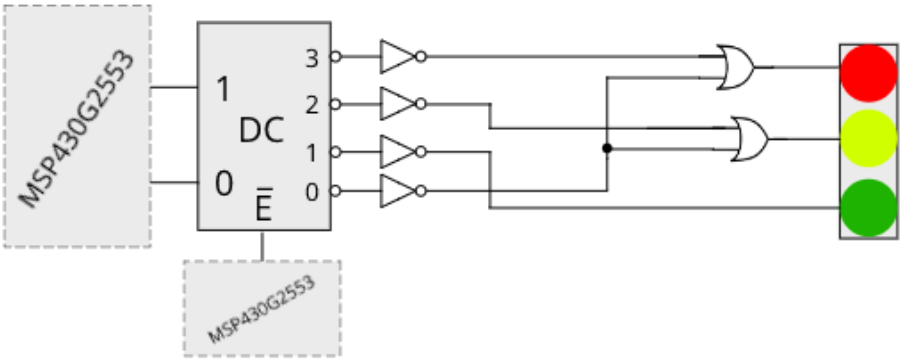
Semafor za pešake je dosta jednostaviji za realizaciju. On ima samo dva stanja pa je potreban jedan dekoder čiji su ulazi tumačeni na sledeći način:

- 0 - Crveno
- 1 - Zeleno

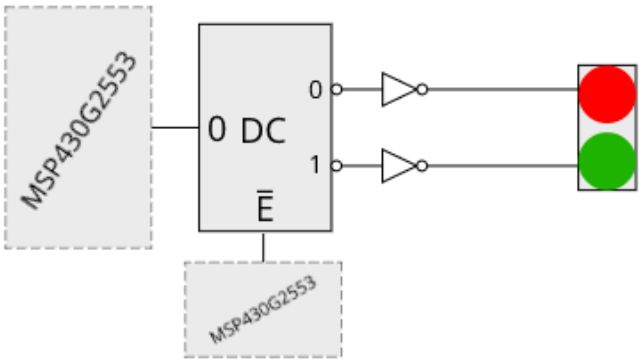
Kako izlazi dekodera daju invertovanu vrednost, potrebno je dodati samo jedan invertor i semafor je spreman za korišćenje.

2.2.3 Strukturne šeme

U nastavku su date strukturne šeme semafora za automobile i za pešake. Treba računati da je potrebno naraviti još jedan primerak tih šema za semafore u suprotnom smeru.



Slika 2.2: *Kombinaciona mreža za semafor za automobile.*



Slika 2.3: *Kombinaciona mreža za semafor za pešake.*

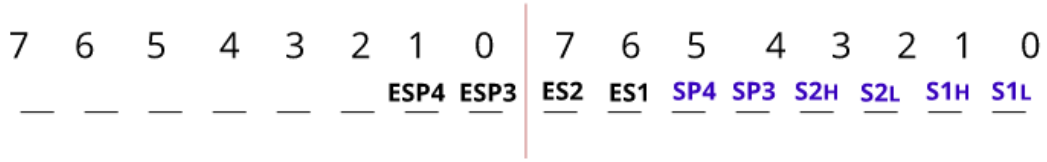
2.3 Programiranje mikrokontrolera

Kada je završena izrada hardverskog dela projekta, potrebno je povezati mikrokontroler sa ulaznim pinovima dekoderima i tako osposobiti paljenje

svetala na semaforima.

2.3.1 Opcode

Opcode predstavlja dvobajtnu "instrukciju" koja sadrži informaciju o tome koji semafor kako treba da radi. Format instrukcije je dat u vidu slike i definisan je na sledeći način:



Slika 2.4: *Prikaz opcode-a koji će parsirati mikrokontroler.*

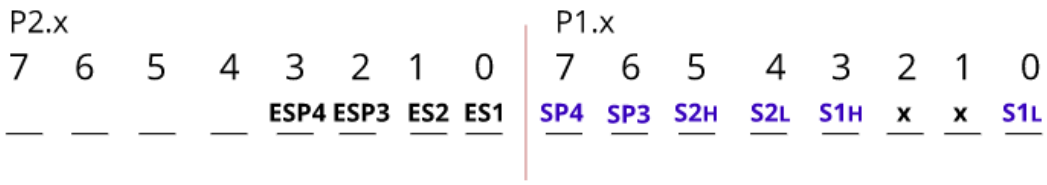
S1, S2, SP3, SP4 predstavljaju semafore sa slike 2.1, a ES1, ES2, ESP3, ESP4 su enable biti semafora S1, S2, SP3 i SP4, respektivno. Indeks $S1_H$ govori da je to viši bit koji se vodi na ulaze dekodera.

2.3.2 Raspored pinova

Potrebno je izvršiti raspodelu i zauzimanje pinova na mikrokontroleru. U okviru njega postoje:

- P1.0 - out
- P1.1 - UART reserved
- P1.2 - UART reserved
- P1.(3..7) - out
- P2.(0..3) - out

Ovo je spisak svih pinova koji su nam bili potrebni za realizaciju projekta. Način na koji smo koristili alokaciju je sledeći:



Slika 2.5: *Prikaz rasporeda pinova na mikrokontroleru MSP430G2553 koji je povezan sa kombinacionom prekidačkom mrežom.*

Enable biti se šalju onako kako zaista želimo da upravljamo semaforima (0 - ugašen, 1 - upaljen) bez obzira na to što je enable bit aktivan kada je jednak nuli (*active low*). Funkcija koja vodi računa o tome je `_enable_semaphore()` (pogledati poglavlje 2.3.3).

2.3.3 Funkcije za lakši rad

Da bi se omogućio lakši rad sa mikrokontrolerom, osmislili smo set funkcija koje mogu da se koriste prilikom programiranja. Sada kada znamo kako su određeni biti raspoređeni, potrebno je samo da baratamo sa pročitanim binarnim vrednostima.
Delimične deklaracije funckija koje olakšavaju rad date su u nastavku:

1. `_enable_semaphore( semID, enable )`
2. `_get_enable_bit_position( semID )`
3. `_set_bits_for_vehicles( action, hbitpos, lbitpos )`
4. `_set_bits_for_passengers( action, bitpos )`
5. `_change_lights( semID, action )`
6. `_parse_rdx( second, first )`

U okviru funkcije `_enable_semaphore( semID, enable )` uključujemo / isključujemo određeni semafor na osnovu njegovog ID-a. Na kombinacionu šemu se pušta invertovana vrednost ***enable*** zbog strukture dekodera

SN74HCT139.

Funkcija `_get_enable_bit_position( semID )` vraća aktivan onaj bit koji odgovara enable bitu u rasporedu bitova na mikrokontroleru (koji smo videli gore - slika 2.5). Na primer, ako se prosledi semID = S2, funkcija vraća 0x2, što odgovara poziciji bita 1 u portu 2. Radi još bolje preglednosti, *svi enable biti su izmešteni na port 2.*

Funkcija `_set_bits_for_vehicles( action, hbitpos, lbitpos )` na osnovu akcije (npr. *action = R (red)*) koja joj se prosledi, menja viši odnosno niži bit u portu P1 i samim tim prosleđuje onu kombinaciju koja je potrebna na ulaze dekodera.

Funckija `_set_bits_for_passengers( action, bitpos )` radi isto što i gornja funkcija ali samo za pešačke semafore.

Funkcija `_change_lights( semID, action )` menja svetlo na određenom semaforu sa zadatim ID-om na osnovu akcije koja može biti R (*red*), Y (*yellow*), RY (*redyellow*), G (*green*).

Funkcija `_parse_rdx( second, first )` se poziva u UART prekidnoj rutini i služi da parsira prvi i drugi bajt dobijen prilikom transporta. Onog momenta kad se prime dve vrednosti, poziva se ova funkcija koja pali određena svetla na semaforima. Prvi i drugi bajt predstavljaju *opcode*. **Napomena**: Parametri su pomereni zbog logičkog rasporeda bita - od najvišeg (sa leve strane) do najnižeg (sa desne strane).

2.3.4 UART

Komunikacija računara i mikrokontrolera MSP430G2553 se odvija preko *USCI UART*-a. Pinovi koji su tada zauzeti su P1.1 i P1.2. Mikrokontroler postavlja prekidnu rutinu u kojoj može da pročita 8-bitni registar **UCA0RXBUF** koji predstavlja bafer u koji se smešta vrednost prenešena sa računara. Kratki insert koda u C-u ove prekidne rutine izgleda vrlo jednostavno:

```
code[ received++ ] = UCA0RXBUF;
if ( received == 2 )
{
   _parse_rdx( code[1], code[0] );
   received = 0;
}
```

Kao što se može primetiti, sa računara se prvo šalje prvi bajt, a potom odma iza njega drugi. Brojač podataka se resetuje vrlo jednostavno onda kada se kružni bafer za podatke `char code[2]` napuni. Tada se i parsiraju bajtovi i pale određena svetla na semaforima.

2.4 Sinhronizacija semafora

Kada je osposobljavanje mikrokontrolera za efikasno korišćenje gotovo, ostaje da se napravi program (aplikacija) koja će da vodi računa o **sinhronizaciji semafora**. Osnovna zamisao jeste da jedan semafor putnog pravca drži crveno dok drugi napravi svoj puni "zeleni" ciklus. Iako korisnik ima puno pravo da menja trajanje svakog svetla na semaforu, automatsko računanje trajanja vremena garantuje da će semafori biti sinhronizovani. Pomoću programskog jezika *Python* i *QtDesigner*-a bili smo spremni da počnemo izradu aplikacije.

2.4.1 Niz naredbi

Kada korisnik unese dužinu intervala koji želi da primeni na raskrsnicu, kreiraju se nizovi sa elemenatima čije vrednosti mogu biti jednake `R, RY, G, BG`. U ovom trenutku uvodimo novu vrednost koju semafor može da ima, vrednost *BG* koja u ovom slučaju označava trepćuće zeleno (engl. *blinking green*). Tačnije, **svaki element niza predstavlja jedan sekund na semaforu**. Na primer, ukoliko je niz `sem1 = [R, R, R, RY, G, G, G, G, ...]` to znači da na semaforu 1 gori 3 sekunde crveno svetlo. Svake sekunde se preko UART-a šalju komande mikrokontroleru. Kada se iscrpi ceo niz, ciklus se završio što znači da ponovo krećemo od početka da brojimo. Tako obezbeđujemo (teorijski) beskonačno dugačku sekvencu rada semafora na raskrsnici.

2.4.2 Thread

Glavni paket koji smo koristili u izradi sinhronizacije semafora jeste zapravo *Thread* paket. Pomoću njega smo vrlo jednostavno pravili niti koje

vode računa o sinhronizaciji. Napravili smo:

1. **regular_timer** thread
2. **fast_timer** thread
3. **transition** thread
4. **yellow** thread

Komunikacija kao i sinhronizacija **između niti** se odvijala upotrebom *sinhronizacione primitive* ***Semaphore***. Ova klasa ima dve metode: **acquire** - ekvivalent *Semaphore.wait()* i **release** - ekvivalent *Semaphore.signal()*.

Set funkcija pisanih u programskom jeziku *Python* treba da pokupi stanja svih semafora u `regular_timer` niti i da od njih napravi opcode koji će biti poslat mikrokontroleru (slika 2.4).

Nit `fast_timer` se koristi onda kada neki od semafora ima naredbu trepćućeg zelenog. Tada se `regular_timer` zablokira i u rad pušta nit `fast_timer` čiji je sleep period skoro trostruko manji od `regular_timer`-a. To znači da se kao efekat dobija da se svetla brže pale/gase tj. da blinkaju.

Nit `yellow` je nit koja naizmenično pali žuto svetlo na svim semaforima za vozila (engl. *toggle*). Rad ove niti označava da semafori nisu u funkciji.

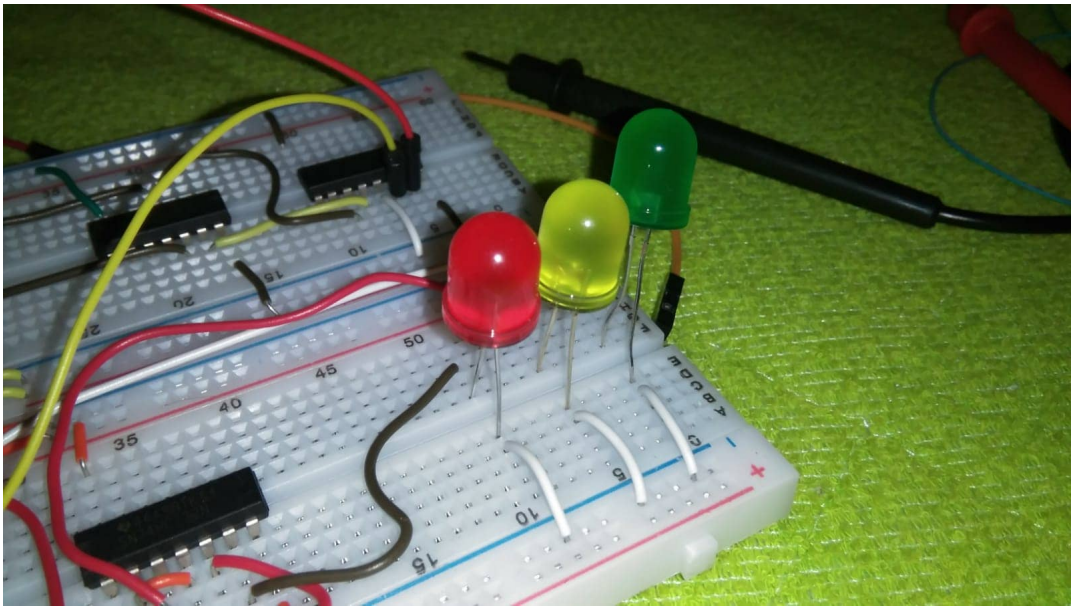
Nit `transition` se koristi u tranzicionom periodu, onda kada je korisnik konfigurisao semafore i pustio ih u pogon. Tada se aktivira ova nit koja par puta upali žuta svetla (blinka žuto svetlo) a potom zadrži žuto svetlo na semaforu dve sekunde. Potom se ova nit zablokira i u rad ponovo pušta nit `regulat_timer` koja ponovo uzima naredbe za svaki semafor na raskrsnici. Ova pogodnost je dodata isključivo da bi se poboljašalo korisničko iskustvo.

2.4.3 Serial

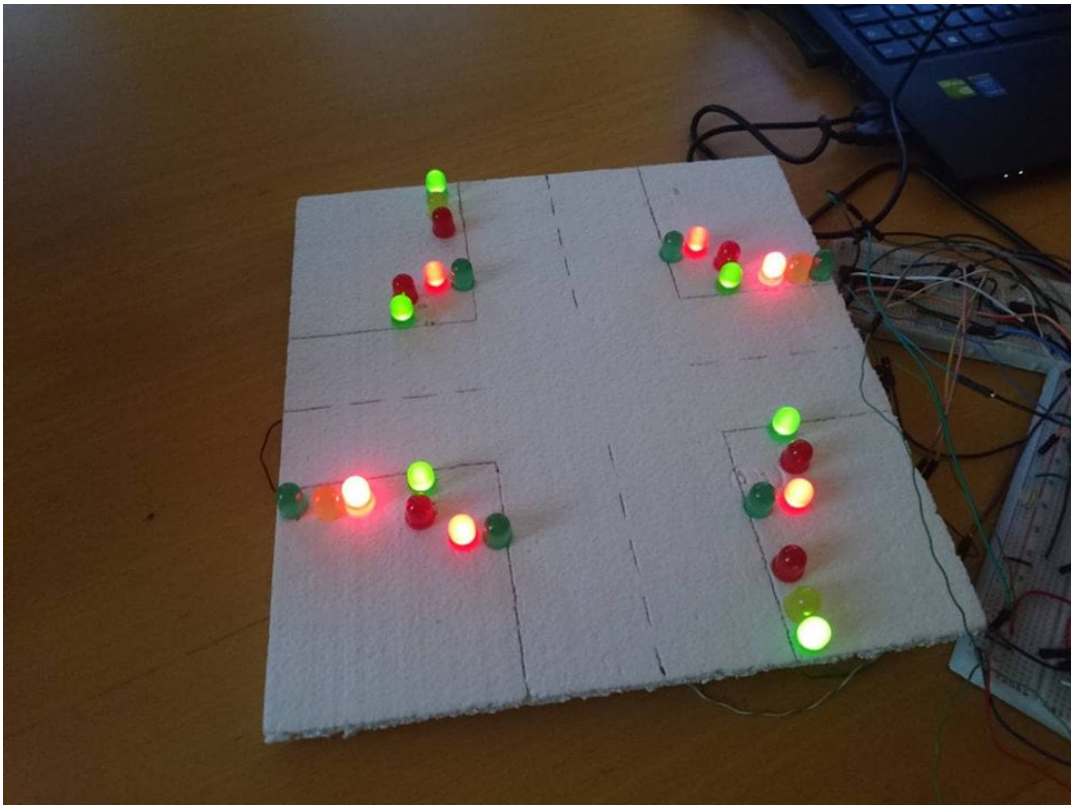
U okviru paketa Serial smo koristili osnovne mogućnosti, kao što su otvaranje komunikacije sa uređajem (*serial.open()*), slanje podataka na serijski port (*serial.send()*) i na kraju gašenje konekcije (*serial.close()*).

2.5 Izgled projekta

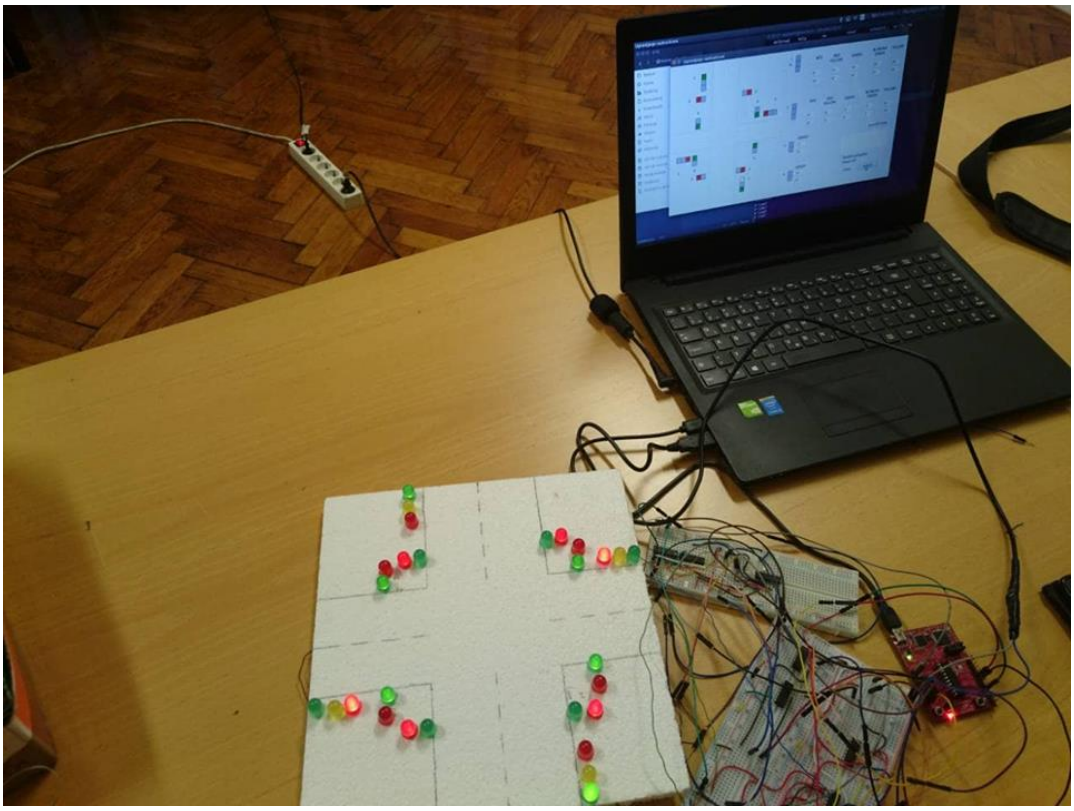
U nastavku prilažemo fotografije gotovog projekta:



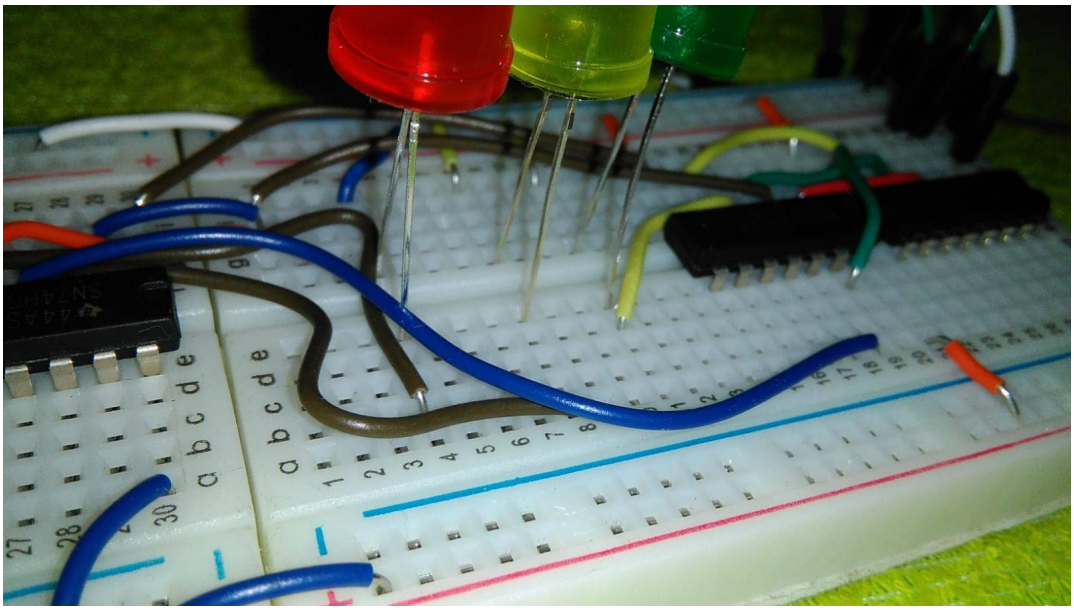
Slika 2.6: *Prikaz dela kombinacione prekidačke mreže za semafor za pešake.*



Slika 2.7: *Prikaz semaforizovane raskrsnice (detalj).*



Slika 2.8: *Prikaz semaforizovane raskrsnice.*



Slika 2.9: *Detalj kombinacione prekidačke mreže.*

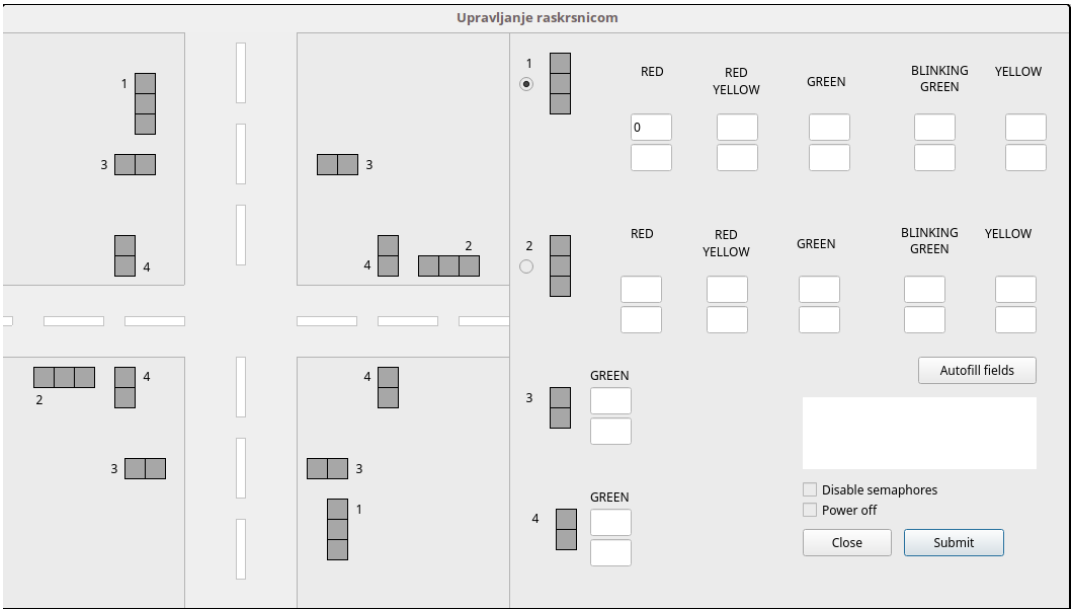
Glava 3

Korisnička aplikacija

Svaki korisnik računara voli da vizuelno ima pregled onoga što želi da uradi. Ako to pak nije slučaj, korisnik najčešće ne zna gde treba da klikne ili šta treba da uradi na računaru da bi postigao željeni cilj. Poznavajući želje samog korisnika, a istovremeno u skladu sa projektnim zadatkom, napravili smo aplikaciju upotrebom *QtDesigner*-a i potom pokrenuli aplikaciju u *Python*-u.

3.1 *GUI* dijagnostika

U okviru GUI dijagnostike, korisniku je omogućeno da proizvoljno postavi vreme (u sekundama) za sve kombinacije svetala: *Crveno*, *Crveno + Žuto*, *Zeleno*, *Trepćuće zeleno*, *Žuto*. Nakon uspešno popunjene vremenske linije prvog semafora, klikom na dugme "*Autofill fields*", aplikacija proračunava najefikasniji opseg vrednosti za semafore, praveći semafore sinhronizovanim. Korisnik sam posle toga može da menja vrednosti na semaforu, a ukoliko naruši sinhronizovanost, program mu neće dozvoliti da nastavi dalje. Izgled aplikacije je dat u nastavku.



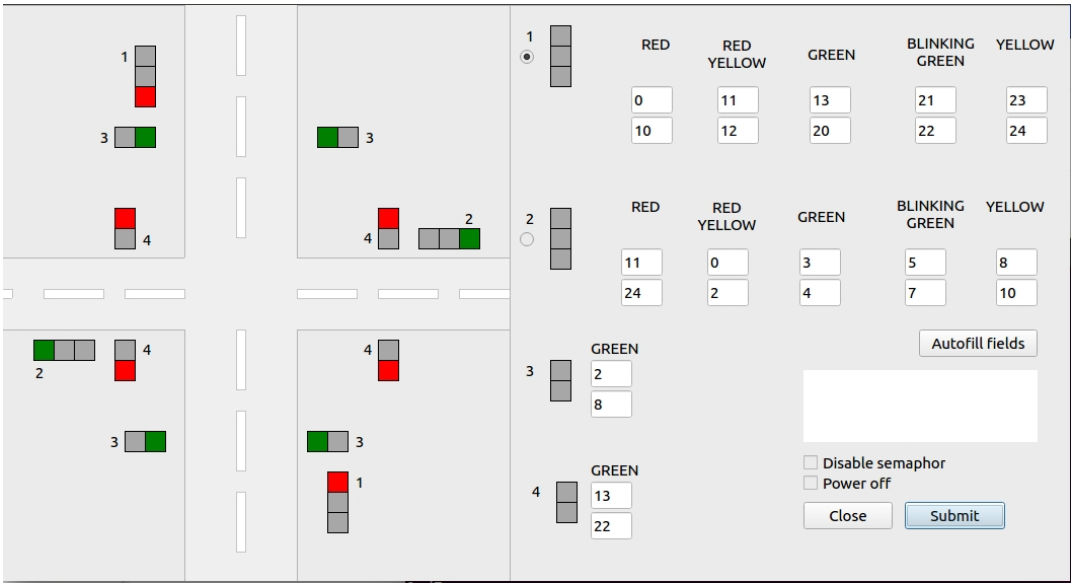
Slika 3.1: *Prikaz korisničke aplikacije.*

3.2 Dodatne opcije

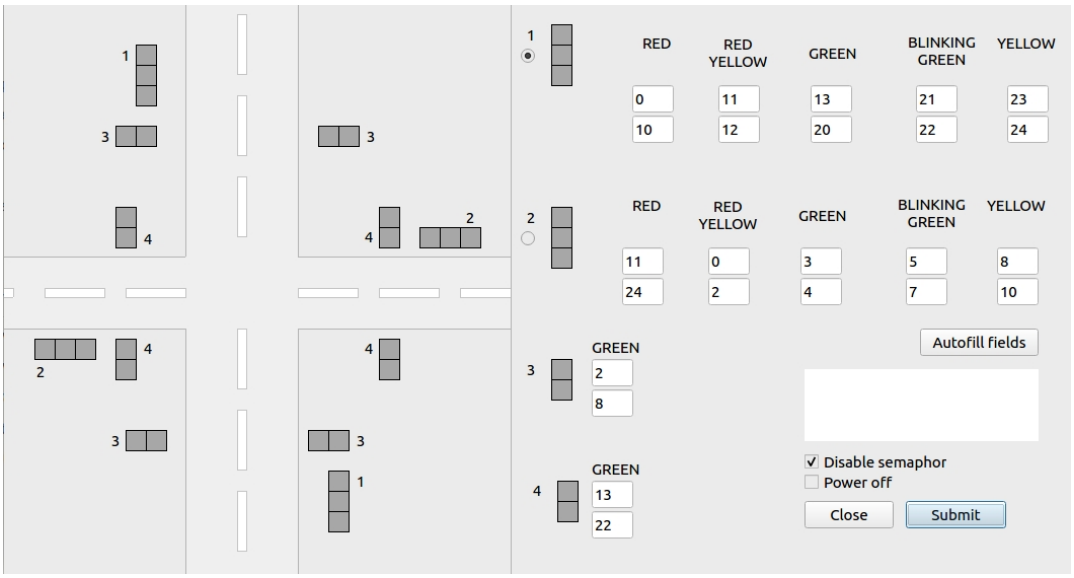
U okviru dodatnih opcija, korisnik može da onesposobi (engl. **disable**) semafore i samim upali trepćuće žuto na njima. Ukoliko pak poželi da ponovo upali semafore, cela raskrsnica će proći kroz *period tranzicije* (pogledati odeljak 2.4.2 na strani 13) i nakon 3-4 sekunde će se uspostaviti aktivno stanje semafora. Korisnik može i da potpuno ugasi raskrsnicu čime će svi semafori biti isključeni.

3.3 Aplikacija ”u akciji”

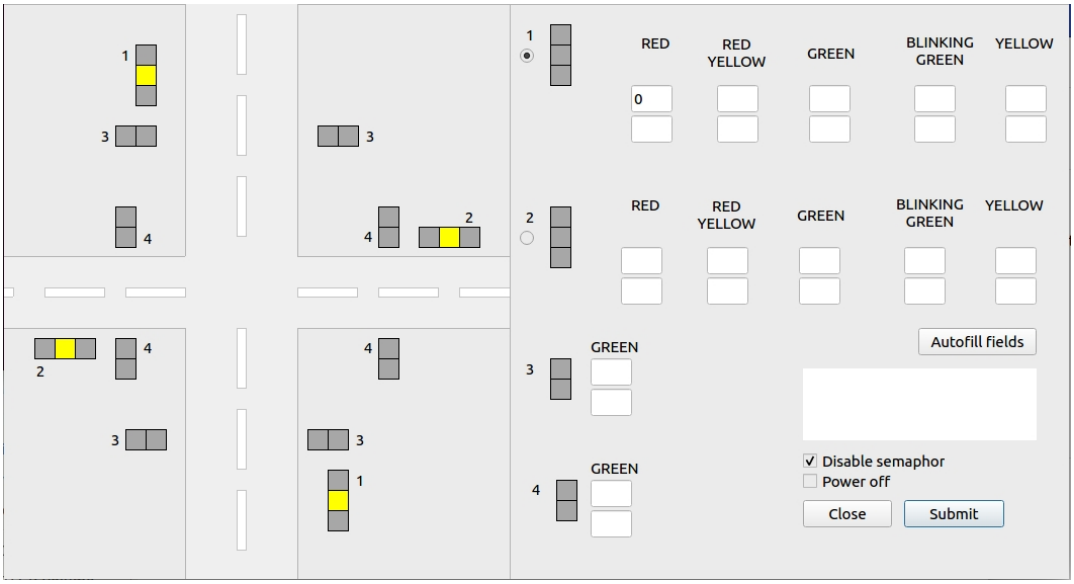
Kada se pravilno unesu vrednosti, rad aplikacije izgleda na sledeći način:



Slika 3.2: *Prikaz aplikacije kada je korisnik konfigurisao vrednosti.*



Slika 3.3: *Konfigurisanje vrednosti u aplikaciji.*



Slika 3.4: *Dodatna pogodnost: Svi semafori su onemogućeni.*

Glava 4

Zaključak

Celokupna izrada semaforizovane raskrsnice sa pravljenjem kombinacione prekidačke mreže proširuje stečeno znanje na predmetu PMT i na zanimljiv način približava studenta konkretnim problemima iz prakse.

4.1 Način razmišljanja i algoritmi

Iako deluje trivijalno, način razmišljanja je veoma bitan kod izrade projekta. Da li ćemo nešto zapisati sa tri bita ili samo sa jednim, dosta menja na planu dalje izrade projekta odnosno omogućava buduće modifikacije. Konkretno, u našem slučaju, uštedeli smo dosta pinova na mikrokontroleru koristeći dekodere i logička kola.
Što se tiče samih algoritama, pre svega mislimo na *pseudokod*, odnosno skrećemo pažnju koliko je važno imati skicu onoga što je potrebno napraviti a potom to isto "sprovesti u delo".

4.2 Prostor za modifikacije

Kao i kod svakog drugog projekta, ostavljen je prostor za modifikacije koje je moguće naknadno implementirati u projektu. Kao jedna od dodatnih modifikacije bi bila recimo sinhronizacija ulice (sa od po dva semafora) koji bi trebalo da smanje gužvu na ulicama. Modifikacija se može dodatno napraviti sa dodavanjem uslovnog zelenog svetla na jednom ili sva četiri skretanja (što na terenu nije tako čest slučaj). Takođe, ono što je zanimljivo jeste da se mogu dodati tasteri za pešake kako bi mogli još brže da pređu ulicu i tako zatvore semafore pre vremena.

Literatura

[1] *http://www.ti.com/lit/ds/symlink/msp430g2553.pdf*, 16.08.2018.

[2] *https://docs.python.org/3.4/library/asyncio-sync.html#semaphores*, 16.08.2018.

[3] *https://pyserial.readthedocs.io/en/latest/pyserial.html*, 16.08.2018.

[4] *http://www.ti.com/lit/ds/symlink/sn54ls04-sp.pdf*, 16.08.2018.

[5] *http://www.ti.com/lit/ds/symlink/sn54ls04-sp.pdf*, 16.08.2018.

[6] *http://www.ti.com/lit/ds/symlink/sn54ls139a-sp.pdf*, 16.08.2018.