Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Praktikum iz merno-akvizicionih sistema

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Faculty of Technical Sciences

Ss. Cyril and Methodius University
Faculty of Electrical Engineering and Information Technologies

Zagreb University of Applied Sciences

School of Electrical Engineering
University of Belgrade

Faculty of Physics
Warsaw University of Technology

Co-funded by the Erasmus+ Programme of the European Union

Lekcija 8

Mašina stanja sa *Event* strukturom, Funkcionalna globalna promenljiva, Programsko upravljanje korisničkim interfejsom, Distribucija aplikacije

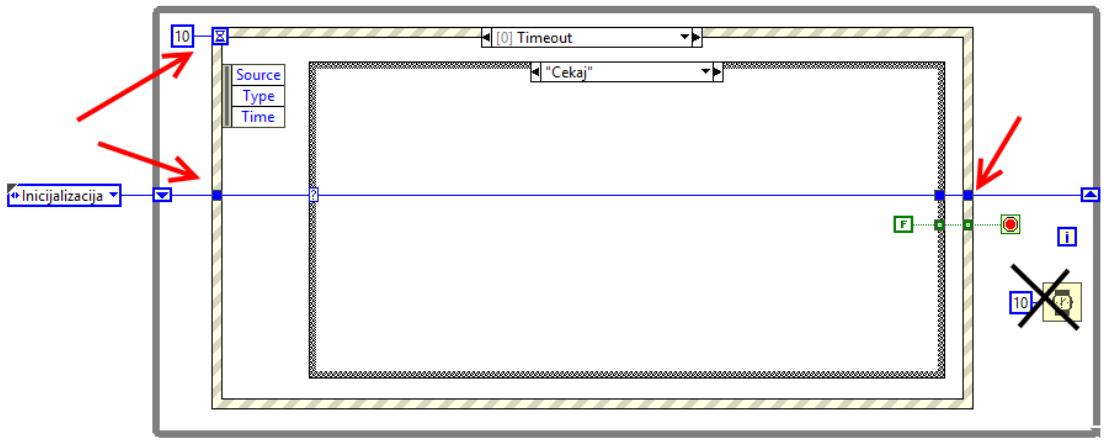
1. Cilj vežbe

Cilj vežbe je da studente upozna sa:

- Mašinom stanja sa *Event* strukturom.
- Funkcionalnom globalnom promenljivom.
- Programskim upravljanjem korisničkim interfejsom.
- Unapređenjenjem postojećih VI.
- Pripremom aplikacije za distribucije.
- Kreiranjem izvršnog fajla (EXE).
- Kreiranjem *installer*-a.

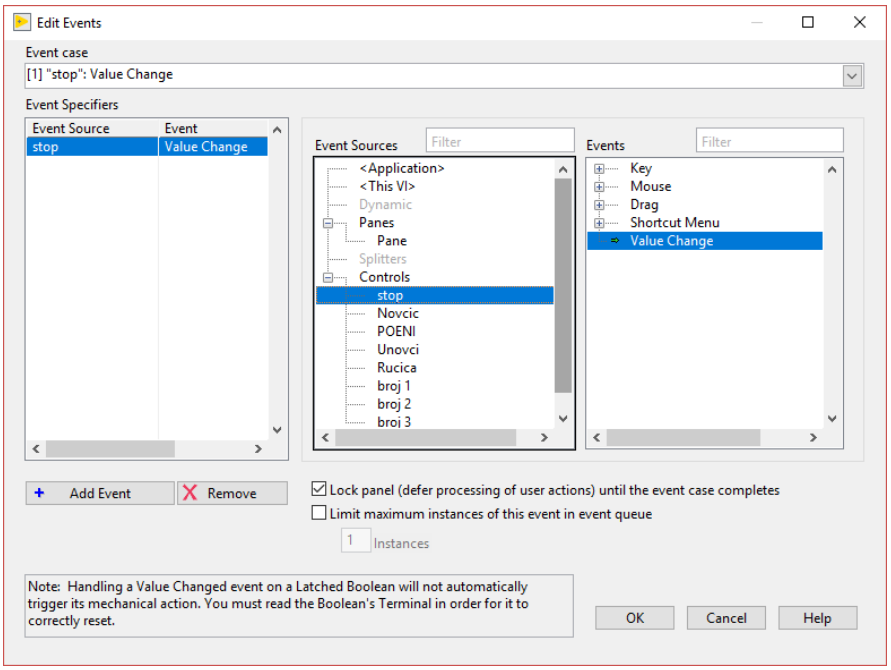
Zadatak 8.1.

Polazeći od zadatka 7.2. realizovati Mašinu stanja sa *Event* strukturom: Napraviti program za igru na sreću. U ovoj igri na sreću povlačenjem ručice na korisničkom interfejsu na dole u članove tročlanog niza brojeva počinju da se upisuju slučajne celobrojne vrednosti od 0 do 100. Obezbediti da se vrednosti menjaju na 10 ms. Vraćanjem ručice u početni položaj se upisivanje vrednosti prekida i trenutno zatečeni brojevi se sabiraju. Ukoliko je rezultat sabiranja veći ili jednak 150, igrač dobija poen. U suprotnom igrač gubi dva poena. Igra se završava kada igrač odluči da unovči svoje poene ili kada izgubi sve poene. Početni broj poena pri pokretanju igre je 5. Obavestiti igrača da je izgubio sve poene i tada se igra vraća u početno stanje gde se očekuje da igrač (novi ili stari) ubaci novčić, pritiskom na taster **novčić**. Do ubacivanja novčića ručica za povlačenje i taster **unovči** su u stanju *disable*. Kada igrač ubaci novčić, taster **unovči** i ručica za povlačenje prelaze u stanje *enable*, a taster **novčić** u stanje *disable*. Ukoliko igrač pritisne dugme **unovči**, dati mu mogućnost da se predomisli. Program se može zaustaviti pritiskom na taster **stop**, ali samo dok je broj poena 0.



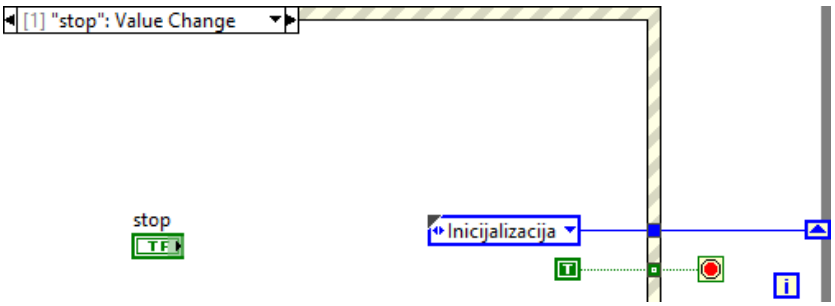
Slika 8.1.

Otvoriti VI “Igra na srecu - pocetak.vi”. Oko *Case* strukture postaviti *Event Structure*, definisati ulaz *timeout* na 10 ms (ne može se ostaviti -1, što znači da će se beskonačno dugo čekati na korisnički *Event*, ali ovde je i zadato vreme zbog generisanja slučajnog broja) i unutar stanja *timeout* povezati ulazno stanje na izlazno za Mašinu stanja (*shift register*). Obrisati *Wait* funkcije, sada je njenu ulogu preuzeto *timeout* ulaz *Event* strukture.
Dodati *Event Case* za taster *stop*, slika 8.2



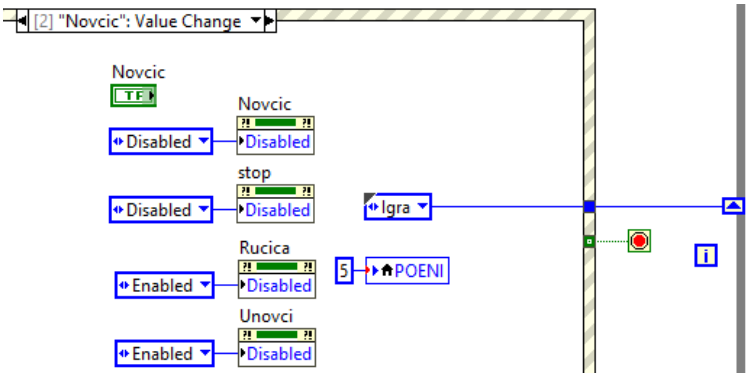
Slika 8.2

Potom, dodati kod kao na slici 8.3 (koristi već postojeći kod, ako nije obrisan).



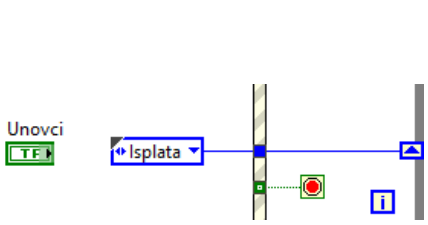
Slika 8.3

Zatim, dodati *case* koji se izvršava prilikom aktivacije taster *Novcic*, slika 8.4.

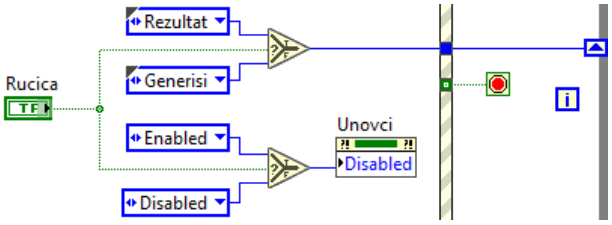


Slika 8.4.

Slično je potrebno uraditi i pri aktiviranju tastera *Unovci* i tastera *Rucica*.

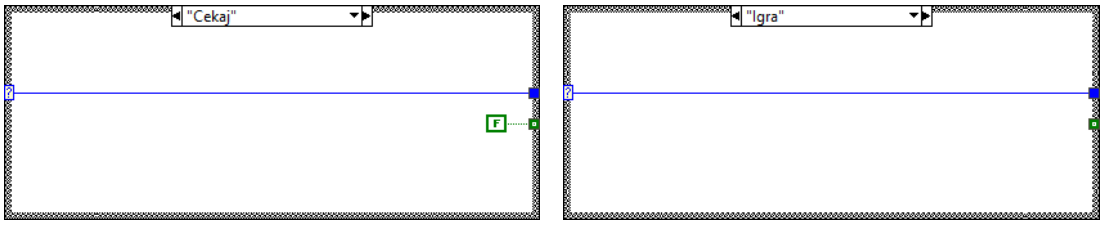


Slika 8.5a.



Slika 8.5.b.

Potom potrebno je prilagoditi i kod unutar *Case* strukture stanja za *Cekaj* i *Igra*, slika 8.6.



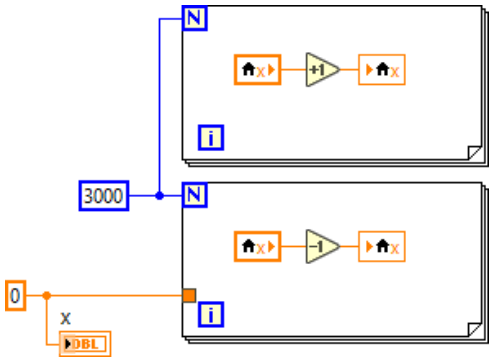
Slika 8.6.

Sačuvati fajl kao "Igra na srecu - kraj.vi"
Testirati funkcionalost početne i krajne aplikacije. Realizacija sa *Event* strukturom se preporučuje, jer početni način provere da li je korisni aktivirao određeni taster (naziva i *pooling*) troši resurse, za razliku od *Event* strukture (*Event driven programming*).

Zadatak 8.2.

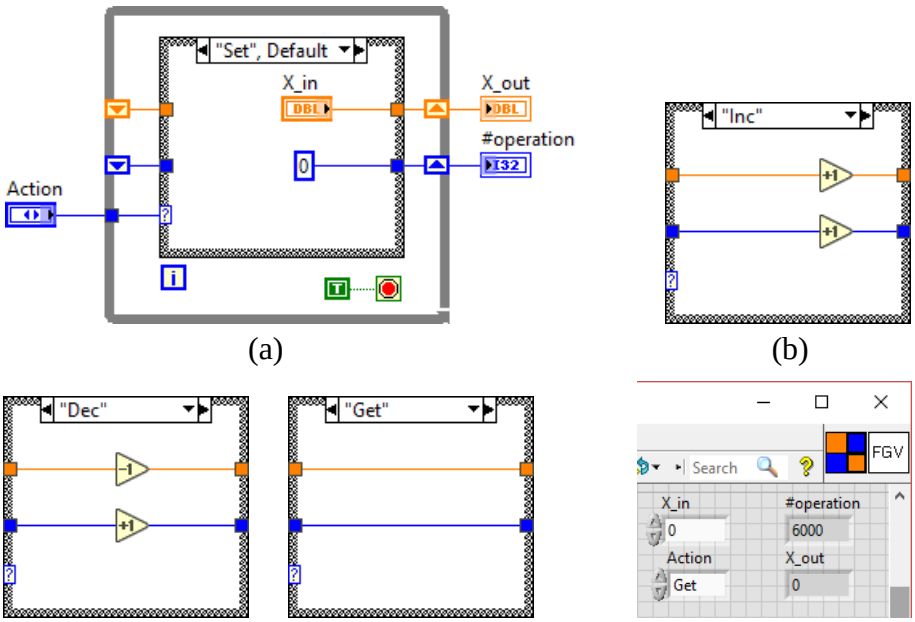
Race Condition se može rešiti i mehanizmom funkcionalne globalne promenljive kojom se vrši komunikacija između dve promenljive.

1. Realizovati kod sa slike 8.7, a potom ga testirati i proveriti koju vrednost sadrži promenljiva *x*.



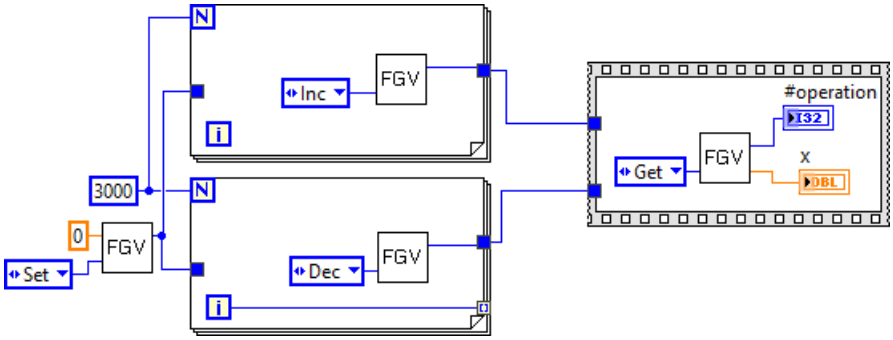
Sl. 8.7.

2. Zatim realizovati funkcionalnu globalnu promenljivu uvođenjem SubVI čija je struktura prikazana na slici 8.8.



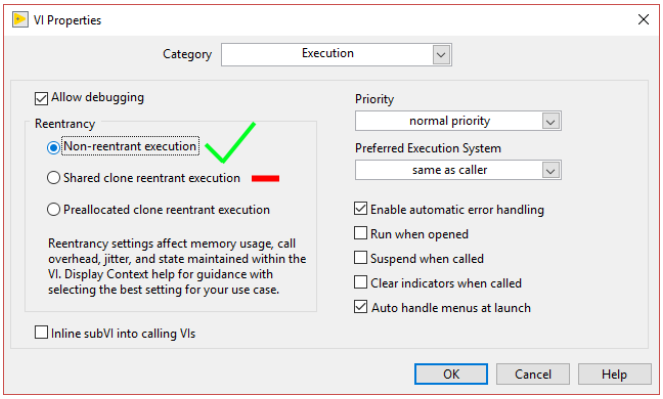
Slika 8.8.

3. Sačuvati realizovani kod pod nazivom “FGV.vi”. Kod sa slike 8.7, modifikovati tako da izgleda kao na slici 8.9.



Slika 8.9.

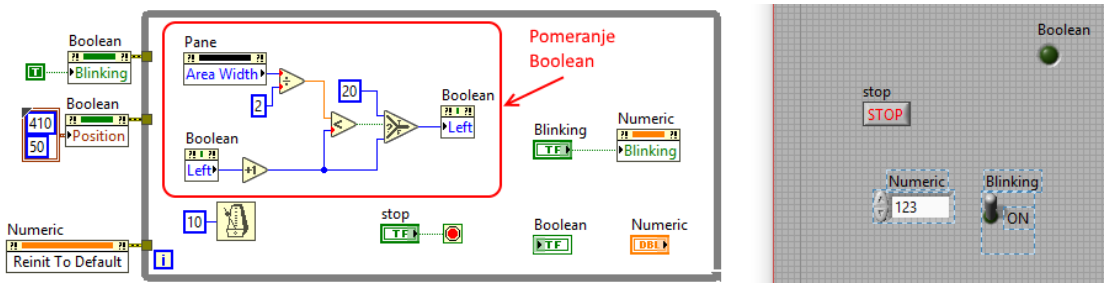
4. Testirati aplikaciju. Ukoliko se ne dobija očekivani rezultat promeniti u *VI Properties* u delu *Executioin*, tako bude izabrano *Non-reentrant execution*, a ne *Shared clone reentrant execution*, slika 8.10. Ukoliko kod radi ispravno, promeniti na *Shared clone reentratn execution* i testirati. **Napomena:** funkcionalna globalna promenljiva koristi *non-reentrant* osobinu funkciju, koja za svaki poziv funkcije koristi isti memorijski prostor, tada se trenutni poziv funkcije ne može započeti sve dok se ne završi prethodni poziv iste funkcije. Sam kod “FGV.vi” sadrži *while* petlju koja se pri svakom pozivu izršava samo jednom i to onaj *Case* koji je zadat pri pozivu funkcije. Neinicijalizovan *Shift* registar omogućuva da se njegova vrednost pamti iz prethodnog poziva.



Slika 8.10.

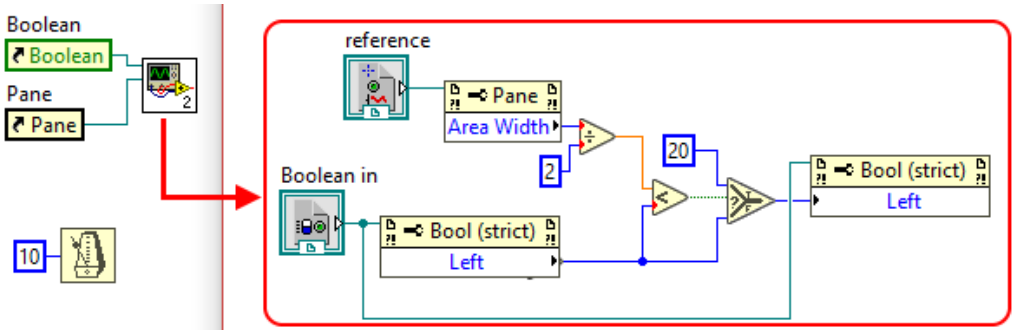
Zadatak 8.3.

1. Realizovati blok dijagram i front panel kao na slici 8.11.



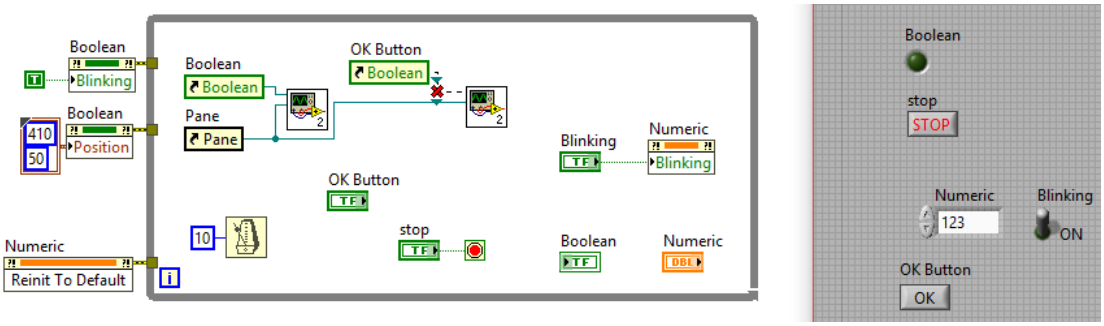
Slika 8.11.

2. Testirati dobijenu aplikaciju. **Objašnjenje:** Pri pokretanju logički indikator *Boolean* pozicionira na koordinatu x = 410 i y = 50, a istovremeno se aktivira i osobina *Blinking* za isti indikator. *Property*-i logičkog indikatora *Boolean* se dobija desnim klikom na terminal *Boolean* i izborom *Create/Propety Node/Blinking*. Pre pokretanja *while* petlje izvršava se i *Invoke Node* za numeričku kontrolu *Numeric* koji je vrća na *default* vrednost (*Reinit to Default*). U samoj *for* petlji aktiviranjem taster *Blinking* osobina *Blinkning* za numeričku kontrolu se pali ili gasi. Levi deo koda pomera logički indikator do polovine širine front panela. Opisani način povezivanja *Property node* (*Invoke node*) sa odgovarajućim terminalom naziva se implicitni.
3. Deo koda označen crvenim pravougaonikom na slici 8.11. selektovati i od njega napraviti SubVI (ići na *Edit/Create SubVI*). Generisaće se kod kao na slici 8.12. Unutar SubVI-a se poziva *Property Node* eksplicitno, prosleđivanjem reference (*control reference*) na određeni objekat u radnoj memoriji. Naime, svakom objektu kao što su logički indikator *Boolean* ili sam front panel (*Pane*) se dodeljuje jedinstveni broj (referenca). Sada *Property node* nije vezan za određeni objekat, već mu se može proslediti referenca na bilo koji objekat koji je tipa *Boolean*.



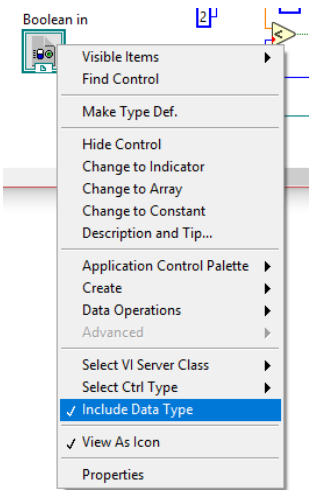
Slika 8.12.

4. Dodati jednu logičku kontrolu na front panel glavnog VI, a zatim dodati još jedan poziv istog SubVI, slika 8.13. Međutim, javlja se greška.



Slika 8.13.

5. Greška je posledica toga što se zahteva da *Property Node* unutar *SubVI* mora biti povezan sa referencom na objekat istog tipa (*strict*). Da bi se uklonila greška, potrebno je deselektovati polje *Include Data Type*, slika 8.14. Sada ne dolazi do greške u glavnom VI. Testirati dobijeni kod. Može se uočiti prednost korišćenja kontrolne reference.

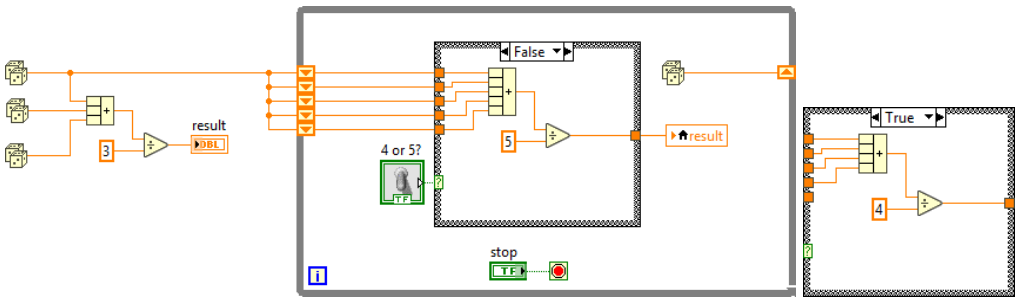


Slika 8.14.

Zadatak 8.4.

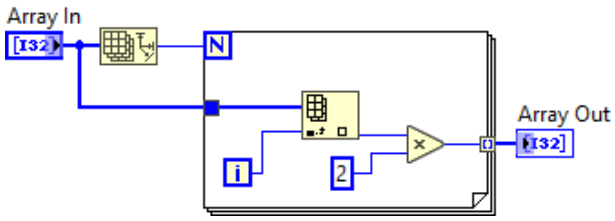
Poboljšanje postojećeg VI.

1. Uklanjanje dupliranog koda. Identičan/sličan kod je potrebno zameniti sa SubVI, jer se tada kod menja samo na jednom mestu, pa je upravljanje kodom jednostavnije. Realizovati kod sa slike 8.15. Zatim obezbediti da se sličan kod ne ponavlja (računanje srednje vrednosti).



Slika 8.15.

2. Uklanjanje komplikovane logike. Realizovati kod sa slike 8.16, a potom identifikovati i ukloniti komplikovanu logiku.

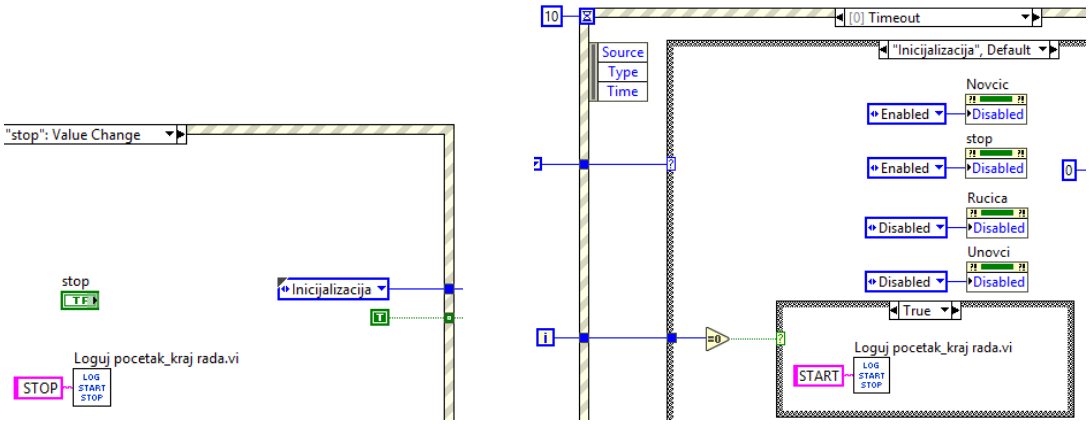


Slika 8.16.

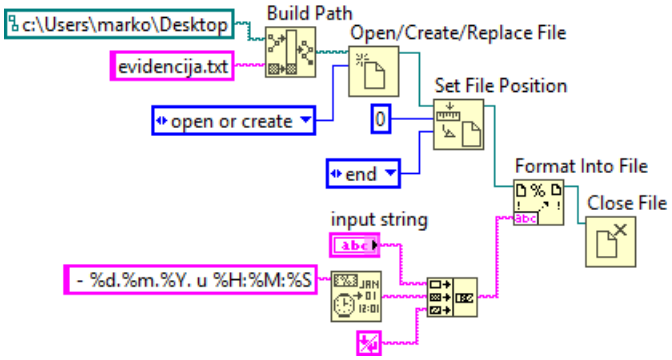
Zadatak 8.5.

Priprema VI za distribuciju kao samostalne aplikacije (*standalone*).

1. Pokrenuti snimljeni VI "Igra na srecu - kraj.vi". Potrebno je obezbediti da se u folderu "c:\Users\marko\Desktop" vodi evidenciju kao log fajl u kojem se upisuje vremenski trenutak (datum i vreme) za svako pokretanje i svaki izlazak iz aplikacije. Napomena: umesto "marko" koristi ime trenutnog korisnika. Realizovati kod kao na slikama 8.17. i 8.18.

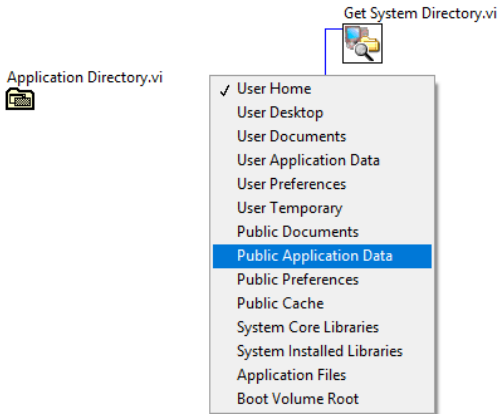


Slika 8.17. Izmena dela koda.



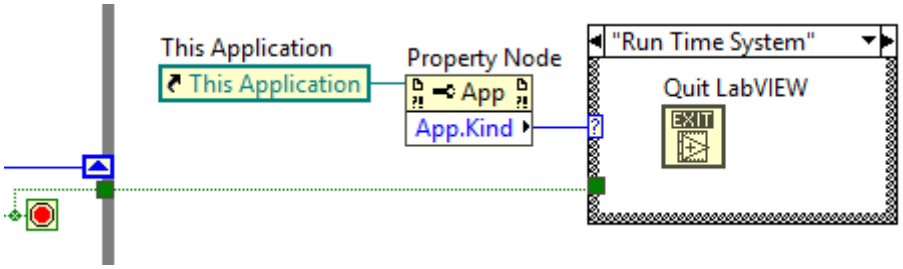
Slika 8.18. Kod za VI "Loguj pocetak_kraj rada".

2. Pokrenuti kod i testirati. Međutim, pri distribuciji ove aplikacije verovatno će se javiti greška, jer je moguće da na računaru krajnjeg korisnika ne postoji zadati folder “c:\Users\marko\Desktop”. Zato je potrebno zadati drugi folder koji sigurno postoji i dve opcije su prikazane na slici 8.19. Zameniti sa “Applicaton Directory.vi” postojeću potanju foldera (slika 8.18).



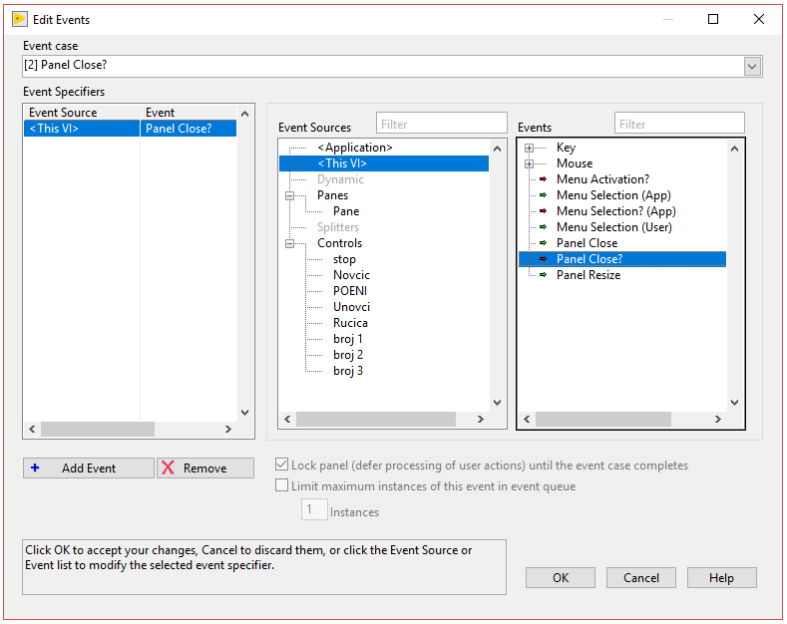
Slika 8.19.

3. Sledeća izmena je neophodna kako bi se osiguralo da se napusti LabVIEW okruženje, prilikom izlaska iz aplikacije. U suprotnom ostao bi front panel koji više ne funkcioniše i korisnik bi morao sam da ga isključi. Potrebno je pozvati funkciju *Exit* kada se aktivira taster *stop*. Međutim, ako se bi se dodala funkcija *Exit* bez dodatnih provera (da li je *standalone*) aplikacija, LabVIEW okruženje bi se isključivalo i u toku razvoja aplikacije, što bi imalo vrlo frustrirajući efekat na korisnika. Da bi se to sprečilo potrebno je dodati kod kao na slici 8.20. Referenca “This Application” dobija se izborom “Vi server reference” sa subpalete *Application Control*. Izlaz iz *Property Node* se vodi na *Case* strukturu. Međutim, *default* vrednost ne sadrži *case* “Run Time System” pa je potrebno desni klik na *Case* strukturu i izabrati *Add Case for Every Value*. Stop iz *while* petlje se dovodi na ulaz *Case* strukture da bi se obezbedilo izvršavanje *Case* strukture nakon *while* petlje.



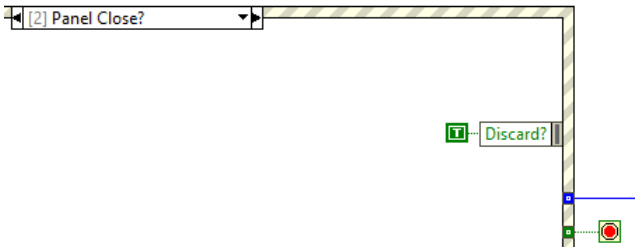
Slika 8.20.

4. Preporučuju se i zabrana zatvaranja aplikacije klikom na X u desnom gornjem uglu, što se može obaviti dodavanjem još jednog *Event*-a u *Event* strukturi, slika 8.21. Potrebno je izabrati događaj *Panel Close?*. Crvena strelica označava da nije u pitanju *Notify Event* već *Filter Event*. Prvi tip događaja obaveštava aplikaciju da se aktivnost već dogodila, dok drugi tip omogućava da se aktivnost i spreči.



Slika 8.21.

5. Zatim, potrebno je dodati i kod kao na slici 8.22. u odgovaraju *case*.



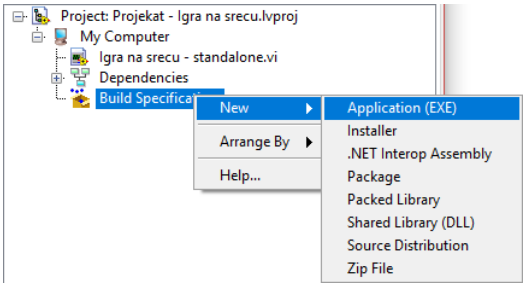
Slika 8.22.

6. Sačuvati dobijeni VI kao "Igra na srecu – standalone.vi". Kreirati novi projekat "Projekat - Igra na srecu" i dodati prethodno dobije VI u njega.

Zadatak 8.6.

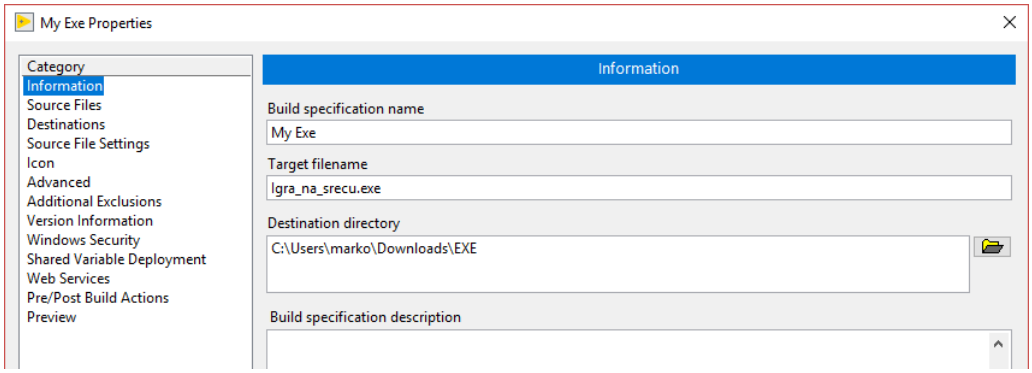
Kreiranje izvršnog fajla (EXE).

1. Desni klik na *Build Specification – New – Application (EXE)*, slika 8.23.

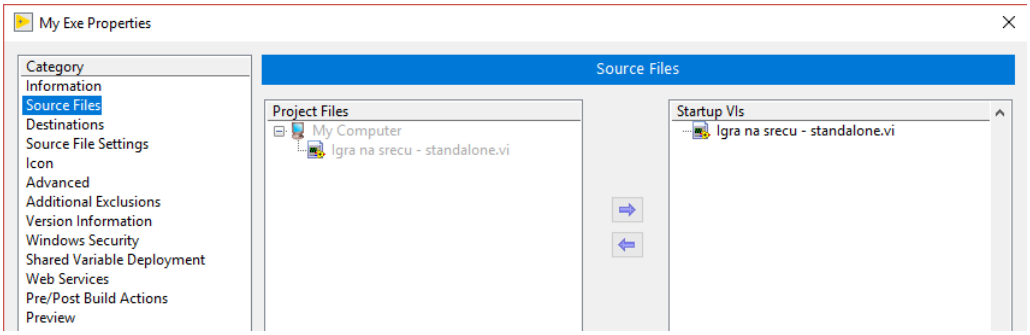


Slika 8.23.

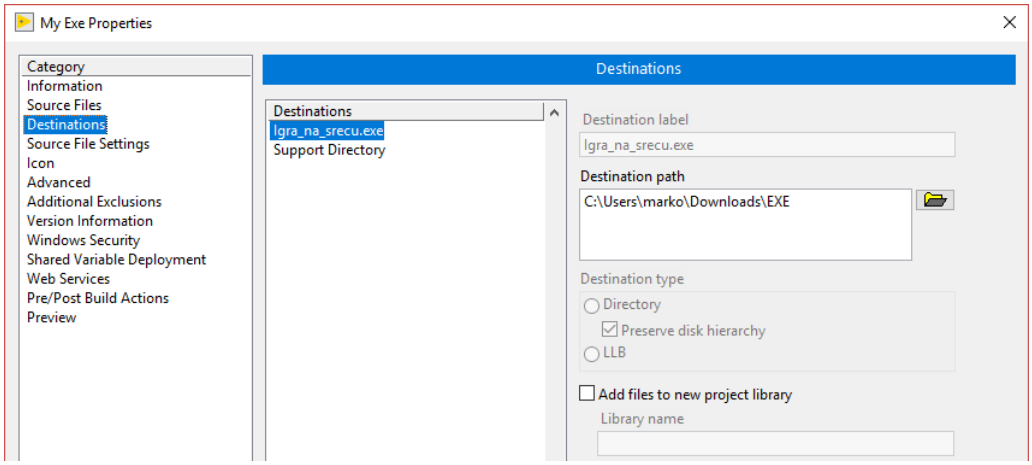
2. Podesiti polja slično kao na slikama od 8.24. do 8.26.



Slika 8.24.



Slika 8.25.



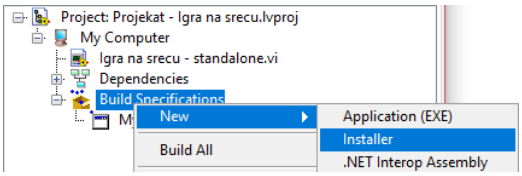
Slika 8.26.

3. Aktivirati taster *Build*. Izaći iz LabVIEW okruženja. Pokrenuti dobijeni izvršni fajl (EXE) i proveriti da li se u istom folderu gde i .exe fajl nalazi i fajl evidencija.txt.

Zadatak 8.7.

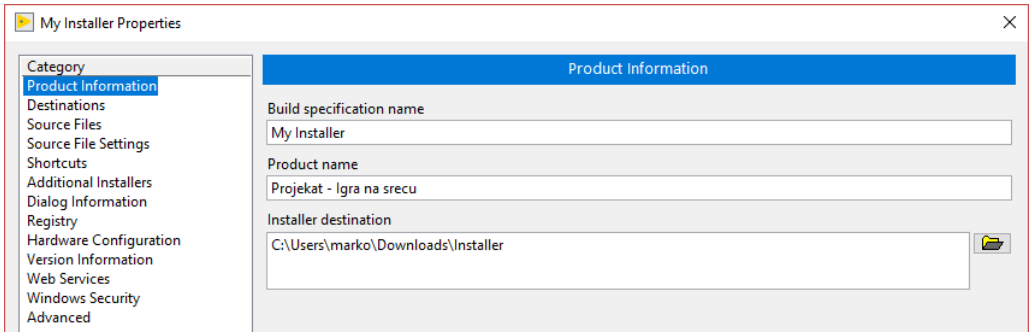
Kreiranje *installer*-a.

1. Desni klik na *Build Specification – New – Installer* slika 8.27.

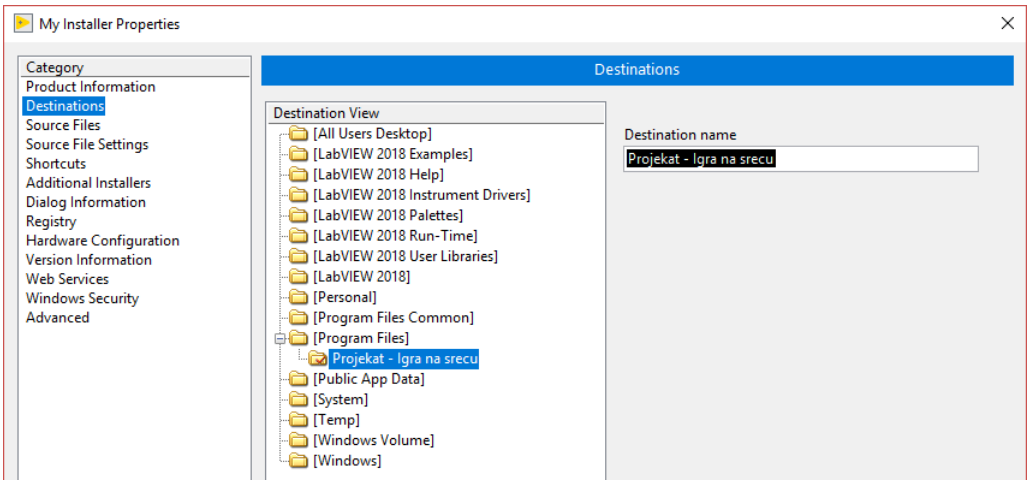


Slika 8.27.

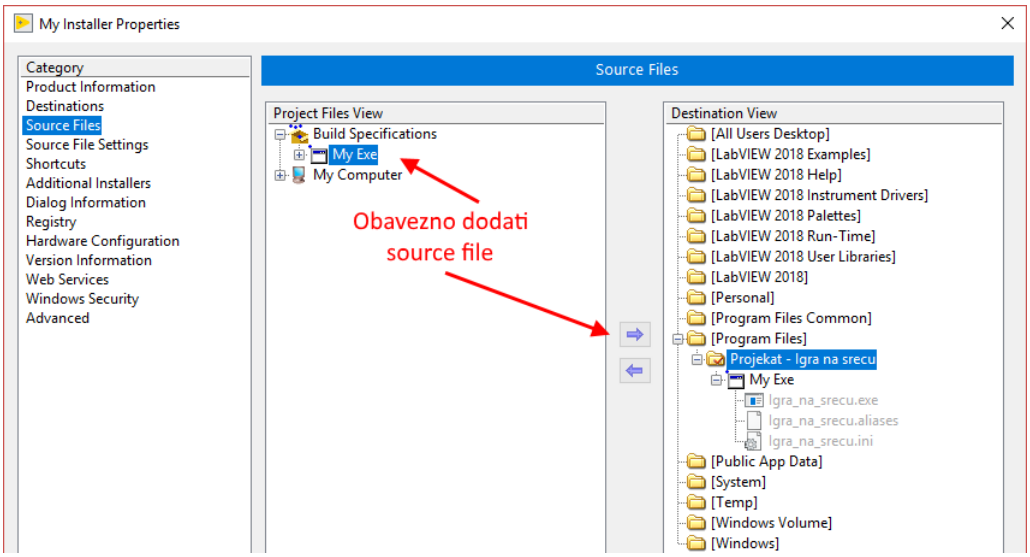
2. Podesiti polja slično kao na slikama 8.28. do 8.30.



Slika 8.28.

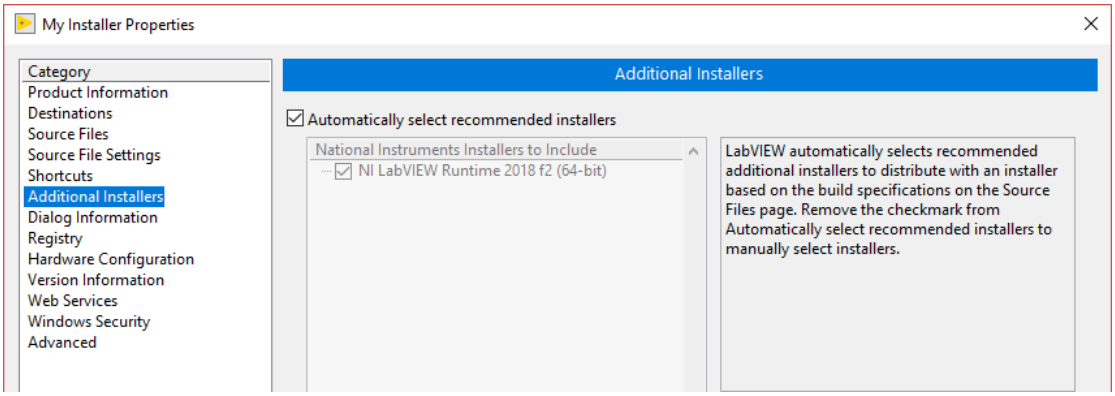


Slika 8.29.



Slika 8.30.

3. Ukoliko se koristi još neko modul za LabVIEW, kao na primer, *NI Vision Development*, potrebno je i *Run-Time* (neophodne izvršne biblioteke) podršku uključiti, slika 8.31, mada bi sam LabVIEW to trebao da uradi.



Slika 8.31.

4. Na kraju potrebno je aktivirati taster *Build* i počinje kreiranje *Installer-a*. Po okončanju postupka proveriti koliko je veličina dobijenog fajla. Ta veličina je posledica ugradnje svih potrebnih delova koje LabVIEW koristi. Bez njih izvršni fajle (EXE) ne bi mogao da radi na računaru koji nema instaliran LabVIEW.