SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJKSIH TEHNOLOGIJA

Sveučilišni diplomski studij

SUSTAV ZA POSREDNO KORIŠTENJE ULAZNIH UREĐAJA  
(System for indirect use of input devices)

Diplomski rad

Luka Kruljac

Osijek, 2021

Page: Izjava o orginalnosti diplomskog rada



Page: Izjavu o pohrani i objavi ocjenskog rada u repozitoriju u sustavu Dabar



#### Sadržaj

[1. Uvod 1](#_Toc79081486)

[2. Postojeća rješenja 3](#_Toc79081487)

[2.1 Usporedba postojećih s novim rješenjem 3](#_Toc79081488)

[3. Razvoj aplikacija na posrednom uređaju 4](#_Toc79081489)

[3.1 WPF Programski okvir 5](#_Toc79081490)

[3.2 MVVM Struktura 5](#_Toc79081491)

[3.3 RAW input 8](#_Toc79081492)

[3.4 Workflow 8](#_Toc79081493)

[3.5 Rezultat i korištenje 9](#_Toc79081494)

[4. Upravljački program za sustave bazirane na UNIX-u 11](#_Toc79081495)

[4.1 Razvojno okruženje 11](#_Toc79081496)

[4.2 Glavna funkcija programa 11](#_Toc79081497)

[4.3 Parsiranje argumenata 11](#_Toc79081498)

[4.4 Dretva za osluškivanje 11](#_Toc79081499)

[4.5 Dretva za izvršavanje 11](#_Toc79081500)

[4.6 Korištenje 11](#_Toc79081501)

[5. Demo projekt 12](#_Toc79081502)

[5.1 Hardware 12](#_Toc79081503)

[5.2 Software 13](#_Toc79081504)

[6. Testiranje 15](#_Toc79081505)

[6.1 Rezultati 17](#_Toc79081506)

[Zaključak 19](#_Toc79081507)

[Literatura 20](#_Toc79081508)

[Popis i opis upotrjebljenih kratica 22](#_Toc79081509)

[Sažetak 23](#_Toc79081510)

[Ključne riječi 23](#_Toc79081511)

[Abstract 24](#_Toc79081512)

[Key words 24](#_Toc79081513)

[Životopis 25](#_Toc79081514)

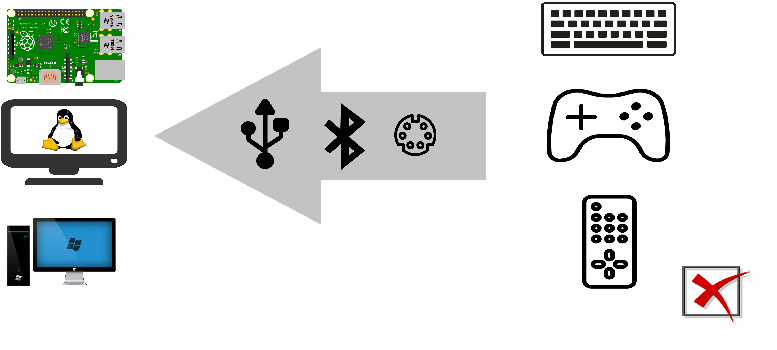
[Prilozi 26](#_Toc79081515)

[Elektronička verzija rada 27](#_Toc79081516)

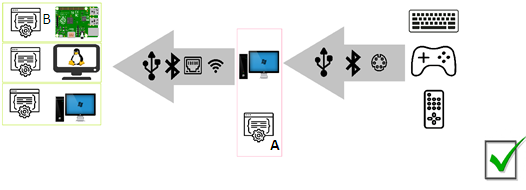
# Uvod

Zadatak ovog diplomskog rada je razviti programsku podršku (eng. software) za posredno korištenje ulaznih uređaja (eng. using input devices indirectly). Software obuhvaća čitavo rješenje (eng. solution) u obliku više projekata koji u cjelini omogućuju korištenje određenog ulaznog uređaja na ciljanom uređaju (eng. target device) bez da su uređaji direktno spojeni. Veza između njih uspostavljena je preko posrednog uređaja koje je sa oba spomenuta uređaja direktno povezan nekom vezom.

Za uspješan rad ovakvog koncepta potrebno je izraditi upravljački software (eng. driver) na strani ciljanog uređaja, odnosno više njih ovisno o platformama ciljanih uređaja koje se žele koristiti te aplikaciju koja se izvršava na posrednom uređaju, a koja za ulogu ima uspostavljenje komunikacijskog puta između ulaznog i ciljanog uređaja odnosno pruža sučelja zaslonskih virtualnih uređaja. Slika 1.1 prikazuje uobičajen način povezivanja ulaznih uređaja PS2/USB/Bluetooth vezom sa ciljanim uređajima, međutim isti nije uvijek moguć te se zadatak ovog rada bavim načinom povezivanja koji je prikazan na Slika 1.2.



Slika 1.1 Uobičajen način povezivanja uređaja



Slika 1.2 Željeni način povezivanja uređaja

Da bi se razjasnilo uvođenje određene kompleksnosti iz prethodnog ulomka potrebno je shvatiti motive ovog projekta. Postoje situacije kada načini povezivanja sa Slika 1.1 nije moguć iz specifičnih razloga. Neki od njih su nabrojani:

* Integrirani ulazni uređaji

Ako se za primjer promatra prijenosno računalo, njegovu integriranu tipkovnicu nije moguće (barem ne na jednostavan način) od spojiti od prijenosnog računala te ju zatim spojiti i koristit na nekom drugom uređaju.

* Ciljani uređaj je mobilan

Ukoliko je ciljani uređaj „na kotačima“, potrebno je pomjerati i ulazni uređaj. Za primjer uzeti učenički projekt automobila pokretan RPI računalom, a koji se upravlja tipkovnicom ili kontrolerom (eng. gampad).

* Ciljani uređaj je dislociran ili fizički nedostupan

Ponekad nije moguće spojiti ulazni uređaj jer smo udaljeni od njega, nemamo fizički pristup, ili ako imamo nemam mogućnost spajanja, primjerice, zauzeti su svi USB portovi

* Ulazni uređaj u fizičkom obliku nije dostupan

Aplikacija na posrednom računalu nudi i tzv. zaslonske uređaje, gdje kliktanje po sučelju simulira pritisak odgovarajuće tipke pa posjedovanje ulaznog uređaja nije nužno.

Ovaj rad obuhvaća cjelokupni proces razvoja sustava, od analize problema, usporede postojećih rješenja, preko razvoja do testiranja. Kroz drugo poglavlje predstavljene su postojoće tehnologije koje imaju sličnu ulogu kao i njihove prednosti i mane u odnosu na koncept sustava iz naslova ovog rada. U poglavljima tri i četiri opisani su postupci razvoja sustava, te funkcioniranje dvije najvažnije komponente sustava:

1. aplikacija na posrednom uređaju, obrađeno kroz treće poglavlje;
2. upravljački program za ciljanu platformu, obrađeno kroz četvrto poglavlje

Također, u sklopu rada pripremljen je i demo sustava čija izrada će bit predstavljena u poglavljupet, dok su rezultati i analiza provedenog testa opisani u šestom poglavlju ovoga rada.

# Postojeća rješenja

Projekt ovog rada po puno toga nije unikata, postoji mnogo rješenja koja obrađuju slične probleme, također za svaki problem nabrojan u poglavlju 1.2 postoji rješenje, no takva rješenja ne rješavaju sve nabrojene probleme u jednoj cjelni ili ih ne rješavaju na efikasan način, odnosno postavljaju dodatne preduvjete koji možda ne mogu biti ispunjeni u svakoj situaciji kada je potrebno upravljati ciljanom platformom.

Primjerice, VNC (eng. Virtual Network Computing) alat iz [2] objedinjuje sve funkcionalnosti koje su obrađene u ovom projektu, štoviše i puno više nego li je obrađeno ovim projektom. Međutim, ovo rješenje ima određena ograničenja po pitanju zahtijeva te ne taj način u određenim situacijama ne može parirati projektu iz ovog rada. Jedan od nedostataka je što ciljani uređaj mora imati spojen ili virtualno montirani displaya, osim toga ovo podrazumijeva isključivo TCP/IP vezu između ciljanog i posrednog uređaja. Osim VNC tu je i SSH (eng. Secure shell) rješenje iz [3] koji također pruža puno više mogućnosti i ima puno širu primjenu, međutim neka ograničenja iz ovog projekta u određenim situacijama daju prednosti projektu iz naslova, štoviše isključivo s SSH protokolom neke od problema iz 1.2 nije niti moguće riješiti. Ograničenje je postavljeno na vezu, SSH za vezu između uređaja podržava TCP/IP i UART. Veći problem ovog alata je što nije moguće koristiti uređaje poput gamepada, barem ne analogni dio, a i pritisak više tipki istodobno nije podržan. Nadalje, aplikacija na strani ciljanog uređaja mora voditi računa o obradi signala s obzirom da SSH prosljeđuje ulaze samo u otvorenom terminalu, a ne čitavom operacijskom sustavu.

## Usporedba postojećih s novim rješenjem

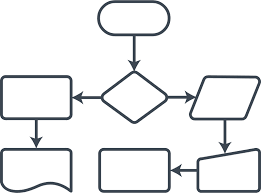
Za razliku od gore spomenutih rješenja, sustav opisan ovim radom pruža mogućnost istovremenog korištenja jednog uređaja na više različitih uređaja neovisno pripadaju li istoj platformi. Za razliku od VNC-a moguće je koristiti i na uređajima koji nemaju ugrađeni display, odnosno moguće je uštedjeti resurse na ciljanoj platformi jer nije potrebno postavljati virtualni display. Također, ukoliko je konkretno skloplje uređaja nedostupno, moguće je koristiti virtualne uređaje, te je programska podrška dizajnirana tako da je u budćnosti moguće dodavati i druge vrste ulanih uređaja. Uspoređujući ovo rješenje s SSH rješenjem prednost je što je krosteći ovaj sustav moguće poslati odzive uređaja različitih od tipkovnice, te je moguće slati više pritisaka tipki tipkovnice istovremeno. Međutim, za postavljanje sustava u određenim slučajevima i dalje je potrebno koristiti SSH onda kada želimo pokreniti upravljački program na ciljanoj platformi.

# Razvoj aplikacija na posrednom uređaju

Pogledom na Sliku 1.2 vidljivo je da u tako postavljenom sustavu posredni uređaj zahtijeva software, komponenta označena slovom A, koji će omogućiti uređaju da postane posrednik između ulaznog uređaja i ciljane platforme. Da bi software odradio svoju ulogu postoji nekoliko zahtijeva među kojima su tri ključna:

1. Osluškivanje strujanja (eng. stream) ulaznih uređaja
2. Osiguranje komunikacijskog kanala s ciljanom platformom
3. Generiranje i slanje poruke kroz komunikacijski kanal

Osim tri ključna zahtijeva, zadani su zahtjevi poput filtriranja određenih uređaja te omogućavanje zaslonskih virtualnih uređaja. Zahtjevi ove komponente kao i njihova međusobna interakcija prikazani su na UML dijagramu na slici 3.1.



Slika 3.1 UML dijagram aplikacije

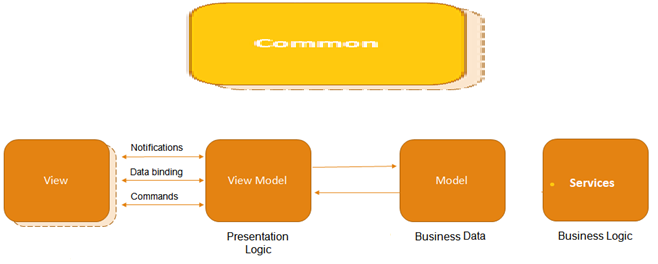
U sklopu ovog prototipa posredni uređaji ograničeni su na uređaje s windows operacijskim sustavom. S obzirom da windows operacijski sustav omogućava grafičko korisničko sučelje (eng. GUI) te uz činjenicu da velika većina korisnika preferira GUI aplikacije u usporedbi s CLI(eng. Comman Line Interface) aplikacijama, logičan izbor za tip ove komponente, odnosno aplikacije koja očekuje interakciju s korisnicima je windows desktop aplikacija.

## WPF Programski okvir

Za izradu windows desktop aplikacije u ovom projektu korišten je programski okvir (eng. framework) otvorenog koda (eng. open source) WPF (eng. Windows Presntation Foundation). Ovaj framwork temeljen je na vektorskom prikazu koji uz široki set značajki pomoću XAML jezika (eng. Extensible Application Markup Language) omogućuje poprilično jednostavnu izgradnju GUI-a neovisnu o rezoluciji, te popratnog pozadinskog koda (eng. backend code) u jezicima iz .NET okruženja (C#, F#, VB). Neke od ključnih značajki WPF jesu da koristi DirectX, odnosno hardversko ubrzanje za iscrtavanje grafičkih elementa, omogučuje povezivanje podataka (eng. data binding) pomoću čega se lako odvajaju cijeline prikaza (eng. view) od cjelina podataka(eng. models). S obzirom da je Miscrosoft autor frameworka, Visaul Studio je zasigurno najbolji IDE (eng. Integrated Development Environment) za razvoj WPF projekta te je on i korišten tijekom razvoja. Od programskih jezika, korišten je C# za backend kao i neke predkompajlirane(eng. precompiled) dinamičke biblioteke koje su razvijene C/C++ jezikom.

## MVVM Struktura

Pri razvoju aplikacije, da bi se odvojili programski blokovi ispravno je koristit neki od design paterna kao što su MVC, MVVM, MVP itd.. Korištenje design paterna olakšava i ubrzava sam proces razvoja, omogućuje lakše testiranje i pojednostavljuje potencijalne izmjene u idućim nadogradnjama. Za ovaj projekt odabrana je MVVM struktura te su uz osnovne MVVM komponente dodane i komponente servisa (eng. Services) te zajedničke komponente (eng. Common). Ovaj set predstavlja čitavo rješenje (eng. Solution) aplikacije, prikazano na Slika 3.2. Komponente Common, Model, Services i ViewModel predstavljaju .NET Framework Class Library projekte čiji produkti su datoteke tipa .dll, dok je View komponenta Windows aplikacija na dnu hijerarhije čiji je produkt izvršna .exe datoteka.



Slika 3.2 Hijerarhijski prikaz projekata

Common projekt sadrži dvije klase koje omogućuju korištenje MVVM strukture te korištenje data bidinga. BaseView.cs sa Kod 3.1 je apstraktna klasa koju nasljeđuju sve klase modela i viewmodel. Nasljeđivanjem ove klase omogućuje se izvršavanje event handlera koji će reagirati na promjene vrijednosti odnosno bit će pozvan u Set metodi svojstava koji se prikazuju na View strani. Ovaj mehanizam omogućuje da svaka promjena svojstava u pozadinskom kodu bude automatski promijenjena i u GUI-u. Način upotrebe prikazan je na DelegateCommand.cs definira objekt korišten unutar onih ViewModela čiji View sadrži neke komande te tako omogućuje povezivanje određene funkcije iz pozadinskog koda s određenom interaktivnom GUI komponentom. Sama datoteka sadrži dvije različite klase koje se razlikuju po tipu odnosno omogućuju povezivanje funkcija void tipa ili bilo kojeg generičkog tipa.



Kod 3.1 Upotreba metode u model klasi



Kod 3.2 Prikaz upotrebe u klasi modela

Unutar model projekta definirane su klase koje opisuju objekte te enumeracije korištene u ViewModel i Services klasama. Konkretno, definirani su modeli mogućih linkova te mogućih ulaznih uređaja te apstraknte kalse koje uređaji odnosno linkovi nasljeđuju. Na primjeru Kod 3.3 vidljivo je kako je definiran konektora kojeg kasnije neka konkretna realizacija konektora i nasljeđuje. Dodavanjem novih linkova komunikacija odnsno konketora potrebno je definirati klasu koja će naslijedi klasu Connector te definirati metode SendMessage i Connect. Ovim je positgnuta jednostavnost uvođenja novih načina povezivanja između posrednog uređaja i ciljane platforme. Na sličan način definirani su i modeli ulaznih uređaja.



Kod 3.3

ViewModel sadrži klase koje opisuju backend pojedinog viewa. Za svaku stranicu odnosno kontrolu koja se može pojaviti u grafičkom sučelju aplikacije definiran je odgovarajći viewmodel. Viewmodel kreiraju modele, pozivaju metode servis klasa na određene aktivnosti u grafičkom sučelju poput otvaranje prozora, pritisak tipke, selektiranje checkboxa i sl..

View dio definira izgled svih prozora, stranica i kontorla na pojedinim stranicama. Također, ovaj projekt je startup čitave aplikacije pa je ondje definirana i startup funkcija, a osim toga unutar klase aplikacije definirana je i veza između pojedinug Viewa te pripadajučeg ViewModela.

Da bi rezultat čitavog rješenja mogo biti dostavljen u jednoj izvršnoj datoteci potrebno je pri izgradnji View projekta u izlaznu izvršnu datoteku ugraditi korištene biblioteke, odnsno i izgrađene izlazne .dll datoteke ostalih projekata. Da bi se tako nešto postiglo dovoljno je koristiti gotove pakete Fedora i Costura unutar View projekta.

## RAW input

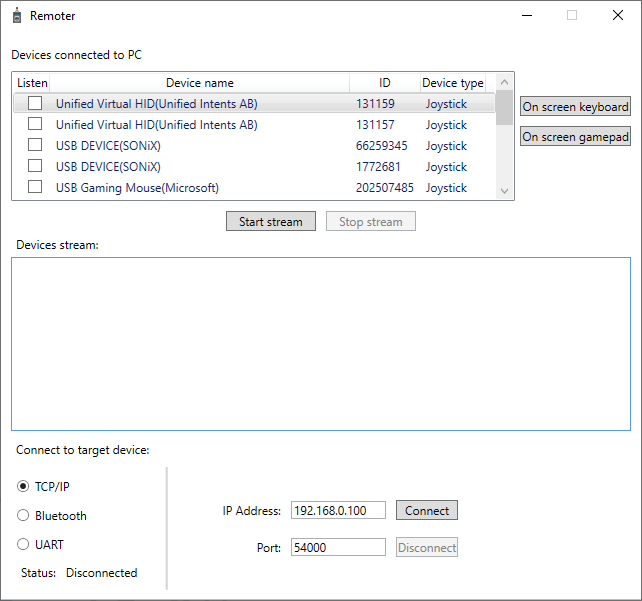
Najvažniji dio ove aplikacije je detektiranje događaja koje kreiraju ulazni uređaji. Za realizaciju ovog zahtijeva korsiteći C# okruženje moguće je koristiti RAW input API. RAW input omogućuje aplikacijama stabilan i robustan način za prihvačanje sirovog ulaza od bilo kojeg korisčnikog ulaznog uređaja(eng. Human Input Device).

## Workflow

Opis UML-a na slici 3.1

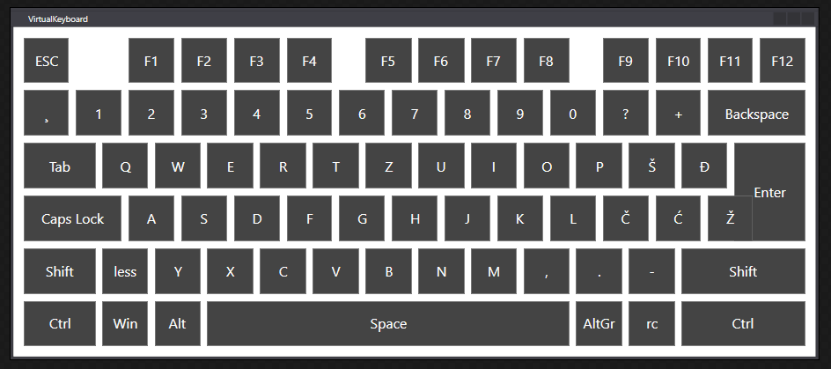
## Rezultat i korištenje

Gradnjom (eng. Building) gore opisanih komponenti konačni rezultat je aplikacija s glavnim prozorom prikazanim na Slika 3.3. U ovom prozoru moguće je uspostaviti link između otvorene aplikacije na posrednom uređaju te upravljačkog programa na ciljanom uređaju. Aplikacija će korisnika obavijestiti o statusu veze te ukolko je veza uspostavljena moguće je krenuti na idući korak. Aplikacija će pri svom pokretnaju izlisati sve pronađene HID uređaje te je na korisniku odabere koji od njih je potrebno proslijediti na ciljani uređaj. Označavanjem uređaja te pritiskom na tipku „Start stream“ svi događaji generirani od strane selektiranih uređaja bit će prosljeđeni prema upravljačkom programu na ciljanom uređaju. Također, moguće je otvoriti i vritualne uređaje poput „On screen keyboard“ i „On screen Gamepad“ vidljivo na Slika 3.4.



Slika 3.3 Glavni prozor windows aplikacije

Slika 3.4 Prozori virtualnih uredaja



# Upravljački program za sustave bazirane na UNIX-u

## Razvojno okruženje

## Glavna funkcija programa

## Parsiranje argumenata

## Dretva za osluškivanje

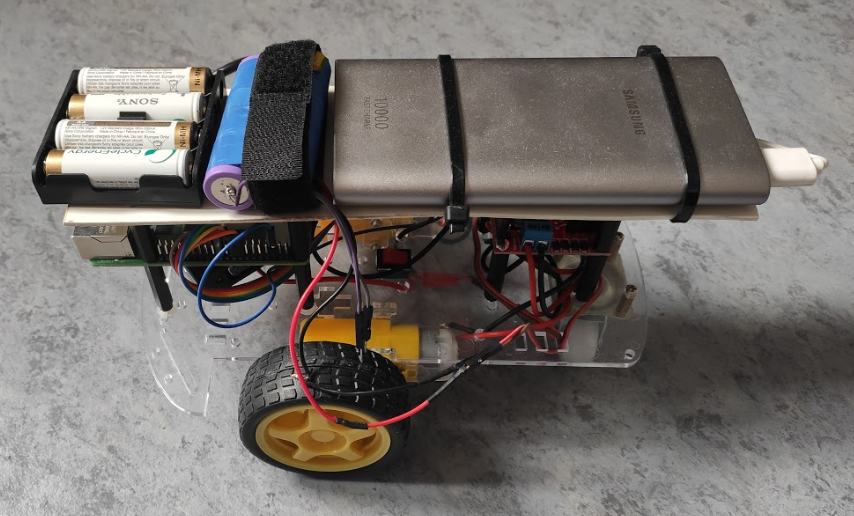
## Dretva za izvršavanje

## Korištenje

Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje thdolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst. Ovdje dolazi neki tekst

# Demo projekt

U skolopu ovoga rada pripremljen je i demo projekat na kojem je prototip i testiran. Ideja demo projekta bila je modelom automobila upravljati pomoću gamepada bez da je isti direktno priključen na sklopljove modela. Za postizanje tog cilja, glavni kontroler ovog modela imao je osiguranu konstantnu internetsku vezu te je putem TCP/IP-a bio povezan s aplikacijom na posrednom uređaju.



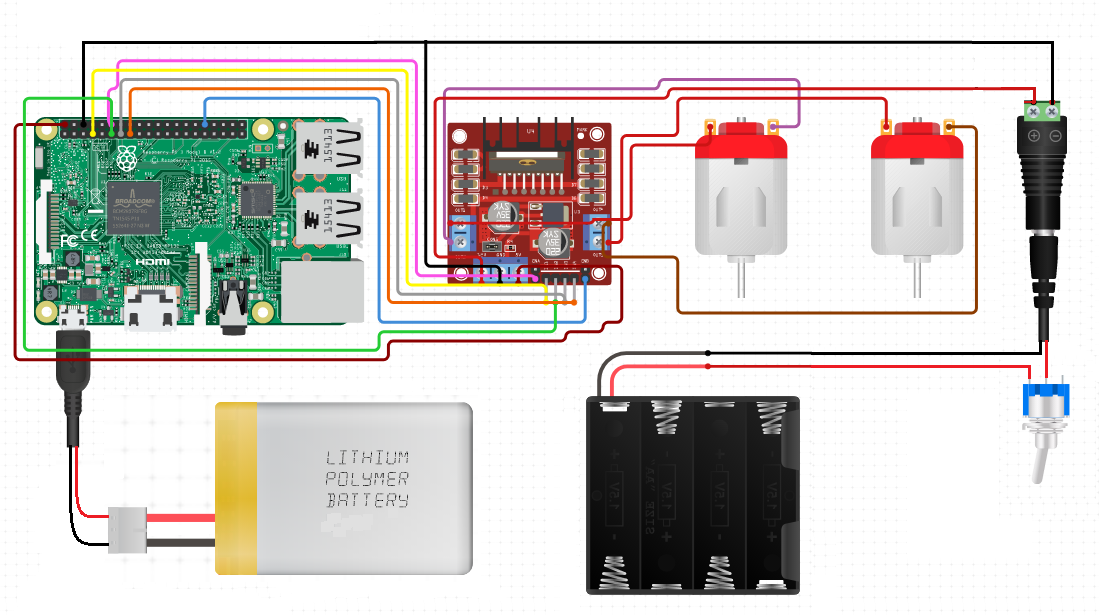
Slika 5.1 Model automobila

## Hardware

Za izradu ovog modela korišten je idući hardware:

* Rasppbery Pi 4
* MicroSD card
* DC motor x 2
* L298N – upravljač DC motora
* 5V Lithium Ion prijenosna baterija s USB portom
* USB A to USB C kabal
* Prekidač
* Battery holder + Lithium Ion punjive baterije
* Vodići
* Plastična konstrukcija s kotačima i odstojnicima

RPI u ulozi glavnog kontroler napajan je prijenosnom 5V Li-Ion baterijom preko USB kabla te ja na taj način omogućena autnomija samog modela. Nadalje, RPI upravlja s L298N kontrolerom te su međusobno povezani s 7 vodića – po jedan za PWM(eng. Puls Width Modulation) signal pojedinog motora te po 2 logička kontakta za određivanje smjera rotacije pojedinog motora, preostali vodić služi za izjednačavanje potencijala u svrhu osiguranja jednakog tumačenja logičke razine na oba kontrolera, odnosno ovaj vodić spaja GND kontakt RPI s GND kontaktom L298N kontrolera. Osim 7 vodića spojenih na RPI, L298N je posredno preko prekidača spojen na izvor koji se koristi za napjanje motora. S obzirom da L298N radi na napjanju između 5V i 12V te da 2 DC motora mogu ostvariti potrošnju preveliku za klasične AAA baterije, za napajanje su korištene dvije 18650 Li-Ion baterije spojene u seriju te tako omogućile napon od 9V te dovoljnu snagu za korištenje 2 DC motora istovremeno na 100% snage. Opis ovog spoja prikazan je na Slika 5.2, te je tako spojen spreman za korištenje, naravno uz pripadajući software.



Slika 5.2 Dijagram ožičenja

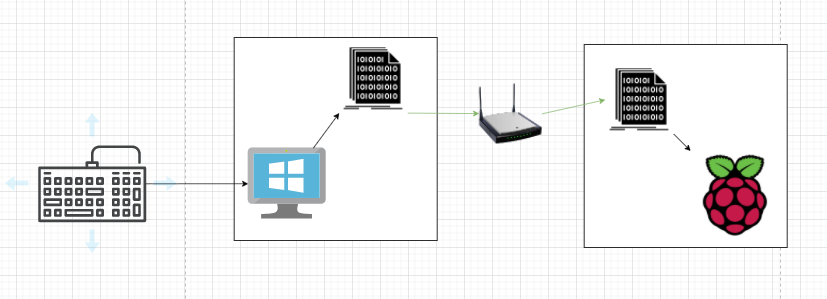
## Software

Sa strane software na ovom modelu situacija je iduća, kao operacijski sustav za RPI mikro računalo instaliran je Raspberry Pi OS također poznat pod imenom Raspbian OS, koji je zapravo verzija Debian operacijskog sustava. Zajedno s ovim operacijskim sustavom instaliran je i python intrepreter. Python je potreban za izvršavanje programa koji upravlja RPI GPIO kontaktima na koje je spojen L298N. Ovaj program pri pokretanju pokušava pronaći ulazni uređaj tipa gamepad te u slučaju uspijeha nastavlja osluškivati strujanje uređaj. Ovisno o podacima koji se pojave u streamu, konkretno položaju analognih usmjerivača, program će izmjenjivati logičke vrijednosti na GPIO kontaktima kako bi odredio smjer i brzinu rotacije pojedinog kotača. Više detalja o ovom programu moguće je pronaći u prilogu pod rednim brojem 2. Naravno, kako bi prototip ovog sustava bio omogućen i na demo projektu, potrebno je instalirati i pokrenuti upravljački program iz poglavlja broj 4.

S obzirom da za software nemam „dijagram ožičenja“, moguće je dodati točne korake kako pripremiti RPI za ovako nešto. Kako instalirat, gdje instlairat, koje verzije pythona itd. – OPCIONALNO, ukoliko ima potrebe

# Testiranje

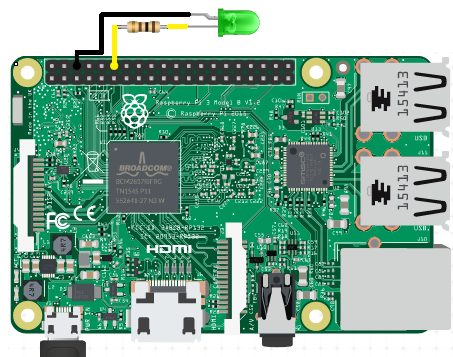
Pri korištenju ovoga sustava vrlo važan parametar je latencija koja se pojavljuje uvođenjem nekoliko dodatnih čvorova između ciljane platforme i ulaznog uređaja. Na Slika 6.1 prikazana je topologija u slučaju korištenja TCP/IP protokola za komunikaciju između ciljane platforme i posrednog uređaja arhitektura. Vidljivo je kako će na letencij povećati vremena potrošena za obradu eventa na posrednom uređaju, zatim upload/download i obrada na routeru te obrada eventa na driveru na ciljanoj platformi.



Slika 6.1 Toplogija hardwareskih i softwarskih jedinica

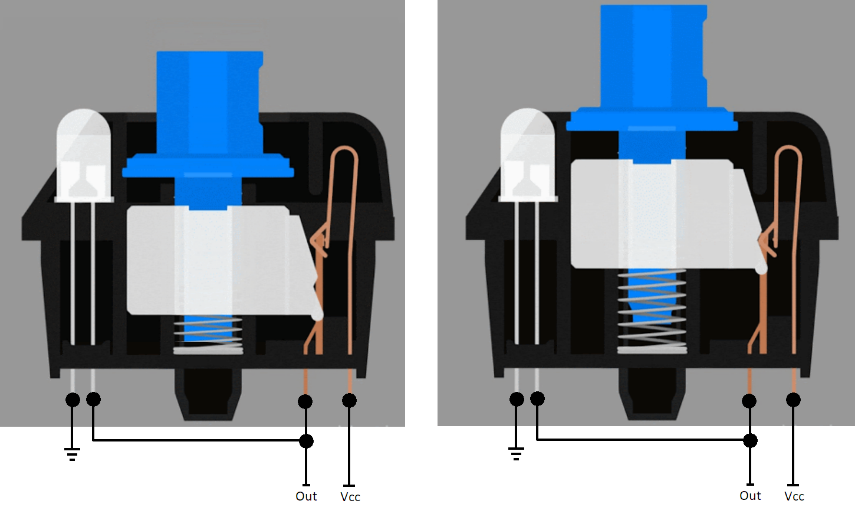
Za testiranje latencije korišteno je primitivno testno okruženje sačinjeno od mehaničke tipkovnice s ugrađenim chery blue swithchevima slični onima sa Slika 6.2, RPI kao ciljana platforma s driverom sustava, LE Dioda spojena na RPI GPIO kontakte, kamera s mogućnosti snimanja slow motion videozapisa, windows računalo s aplikacijom sustava, video preglednik.

Na ciljanoj platformi pokrenut je program, odnosno python skripta koja će pri pritisku neke od tipki reagirati tako što će jedan od GPIO pinova postaviti u stanje high. Na spomenutom pinu je spojena anoda LE diode, dok je katoda LE diode preko otpornika spojena na ground RPI kontakata. Ovim je postignuto da pritiskom tipkovnice LE Dioda zasvijetli.



Slika 6.2 Prikaz LE diode na RPI

Blue chery switchevi s ugrađenim pozadinskim osvjetljenjem također će zasvijetliti pri pritisku tipke. Oba zasvijetljenja, promatranjem golim okom se događaju u istom trenutku, međutim realnost je da ipak postoji vremenski razmak, odnosno kašnjenje LE Diode spojenu na RPI kontakte u odnosu na onu ugrađenu u tipku tipkovnice.



Slika 6.3 Prikaz chery blue switcheva

Da bi se izmjerilo kašnjenje koristi se kamera s mogućošću snimanja u slow motion modu. Poželjno je imati što veći FPS(eng. Frame per second), kako bi mjerenje bilo preciznije. Potrebno je namjestiti kameru tako da se tokom snimanja u sceni nalaze obje LE diode. Nakon što je snimljeno nekoliko testnih snimaka, natavak analize se vrši u nekom od alata za pregled i uređivanje video materijala, primjerice VLC media player. Pomoću video preglednika potrebno je pronaći indeks framea u kojem LE dioda tipke prvi puta zasvijetli, te frame u koje LE dioda spojena na RPI prvi puta zasvijetli. Znajući FPS kamere iz specifikacije, ili izračunavanjem iz podataka video materijala prema izrazu 6.1, gdje je N ukupni broj frameova u videu, a t ukupno vrijeme trajanja videa u sekundama.

|  |  |
| --- | --- |
|  | (6.1) |

Sada, kad je FPS poznat, vrijeme između dva trenutka na dva različita framea lako se izračuna prema izrazu 6.2, gdje je index framea u trenutku , a index framea u trenutku .

|  |  |
| --- | --- |
|  | (6.2) |

Također, treba uzeti u obzir da korištenje tipkovnice u direktnoj vezi također ima određenu latenciju. Naime, tipkovnica ima definirani polling rate, odnosno frekvenciju kojom CPU provjerava podatke na USB sabirnici, što znači da pri pritisku tipke prođe neko vrijeme dok ono bude registrirano od strane CPU-a. Osim pulling ratea, tu je matrix scan rate, odnosno frekvencija mikroprocesora same tipkovnice koji skenira čitavu tipkovnicu i ovisno o stanju zapisuje podatke na sabirnicu. Prema tome za određivanje latencije koju uvodi sustav potrebno je ponoviti isto mjerenje pri direktnoj vezi tipkovnice i RPI. Konačni iznos uvedene latencije može se iskazati prema 6.3, gdje je latencija u sustavu za posredno korištenje ulaznih uređaja, latencija u direktnom sustavu.

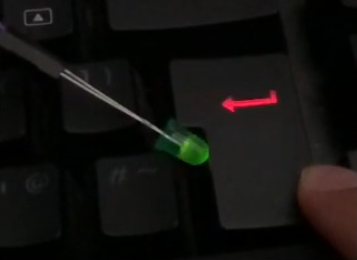
|  |  |
| --- | --- |
|  | (6.2) |

## Rezultati

U Tablica 6.2 prikazani su dobiveni rezultati opisanog mjerenja pri direktnoj vezi tipkovnice i ciljane platforme, dok su u Tablica 6.3 prikazani rezultati mjerenja kada je tipkovnica spojena pomoću sustava za posredno korištenje ulaznih uređaja. Za bolje razumijevanje tablica mjerenja na Tablica 6.1 prikazana je legenda.

|  |  |  |  |
| --- | --- | --- | --- |
| **Indeks okvira** |  |  |  |
| **Trenutak** |  |  |  |
| **Opis** | Okvir neposredno prije pritiska tipke | Okvir u trenutku pritiska tipke. Prvi okvir sa svijetlećom LE diodom tipke i ne svijetlećom LE diodom spojene na ciljanu platformu | Okvir u trenutku detekcije pritiska tipke na ciljanoj platformi. Prvi okvir sa svijetlećom LE diodom tipke i LE diodom spojene na ciljanu platformu |

Tablica 6.1 Legenda

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 240 |  |  | | 240 |
|  | 433 |  | | 316 |
|  | 439 |  | | 332 |
|  | 25 ms |  | | 66.67 ms |
| **Tablica 6.2** Rezultati latencije - direktni spoj | |  | | Tablica 6.3 Rezultati latencije – spoj preko sustava za posredno korištenje | |

Kombiniranjem rezultata iz Tablica 6.2 i Tablica 6.3 te uvrštavanjem u izraz 6.2 ukupna latencija sustava iznosi . Na primjeru ove opreme, uvedena latencija dodaje dodatnih 65% vremena, što rezultira ukupnom latencijom od 66,67. Ova brojka je daleko od idealnog uspoređujući s generalnim rezultatima mehaničke tipkovnice, međutim uzimajući u obzir da postoje tipkovnice koje u direktnoj vezi imaju latenciju od 60 ms. Također, autori znanstvenog istraživanja [5], navode da je frekvencija tipkanja ograničena i iznosi približno 7 HZ, što je približno 130 ms između dva pritiska, pa se može reći da ovaj sustav unosi prihvatljivu latenciju.

Zaključak

Pri donošenju zaključka o ovome sustavu treba naglasiti kako je isti stvoren u domeni diplomskog rada jednog studenta te je produkt svega samo prototip s temeljnim elementima pa prema tome sustav u ovoj fazi nije na razini spremnoj za komercijalnu upotrebu.

Rezultati

Tijekom izrade ovog rada vođeno je računa o arhitekturi softwarea na obje komponente, te su time postavljeni temelji za lakše unapređenje i ubacivanje novih pogodnosti kao što su novi načini komunikacije ili novi tipovi uređaji.

Literatura

1. VNC User Guide  
   dostupno na <https://www.realvnc.com/en/connect/_downloads/VNC_User_Guide.pdf>   
   zadnja posjeta na 28.11.2020.
2. SSH dokumentacija  
   dostupno na <https://www.ssh.com/ssh/protocol/>  
   zadnja posjeta na 28.11.2020.
3. Microsoft WPF dokumentacija   
   dostupno na https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-5.0  
   zadnja posjeta na 20.06.2020.
4. Mateljan, Vladimir ; Đambić, Goran ; Drenovac, Sergej; „New Languages and Technologies for Delivering Learning and Multimedia Contents – XAML and WPF“ 29.7.2008.
5. Duprez. J.; Stokkermans. M; Drijvers, L; Cohen. M; „Synchronization between keyboard typing and neural oscillations“ 25.8.2020.  
   dostupno na https://www.biorxiv.org/content/10.1101/2020.08.25.264382v1
6. Uinput
7. RAW input  
   dostupno na <https://docs.microsoft.com/en-us/windows/win32/inputdev/about-raw-input>  
   zadnja posjeta na 20.06.2020.
8. RPI
9. Linux
10. C99 standard
11. C# 7.0 dokumentacija
12. L298N data sheet
13. Visual studio dokumentacija
14. Argp dokumentacija

Popis i opis upotrjebljenih kratica

|  |  |
| --- | --- |
| VNC | Virtual Network Computing |
| SSH | Secure Shell |
| RPI | Raspberry Pi |
| TCP/IP | Transimsion Control Protocol/Internet Protocol |
| WPF | Windows Presnetation Foundation |
| MS | Microsoft |
| VS | Visaul Studio |
| CLI | Comand Line Interface |
| GUI | Graphical User Interface |
| XAML | Extensible Application Markup Language |
| IDE | Integrated Development Environment |
| MVC | Model View Controller |
| MVVM | Model View ViewModel |
| MVP | Model View Presenter |
| FPS | Framse Per Second |
| API | Application Programming Interface |
| HID | Human Interface Device |
| GPIO | General purpose input output |
| PWM | Puls Width Modulation |
| CPU | Central Procesor Unit |

Sažetak

U današnje vrijeme moderan čovjek okružen je velikim brojem raznih uređaja, tehničkih i operacijskih sustava. Većina takvih sustava dizajnirana je da u nekim trenutcima zahtijeva interakciju s korisnikom. Kako bi se takva inerakcija i ostvarila dizajnirani su posebni uređaji, koji se međusobno razlikuju ovisno o svojoj namjeni, a koji se grupno nazivaju ulaznim uređajima. Činjenica je da u velikom broju slučajeva, kako bi se sustavom upravljalo, nužno je posjedovati ulazni uređaj te biti u neposrednoj blizini samog sustava, što za posljedicu ima da se u okruženju korisnika gomilaju ulazni uređaji. Kako bi se reducirao broj ulaznih uređaja u okruženju korisnika, omogućilo udaljeno upravljanje te omogučilo upravljanje čak i kada ne posjedujemo ulazni uređaj osmišljen je sustav za posredno korištenje ulaznih uređaja. Dizajn prototipa ovakvog sustava opisan je kroz ovaj rad. Obuhvaćene su faze razvoja programske podrške za dvije glavne komponente, aplikacija na posrednom uređaju te upravljački program na ciljanoj platformi. Uz to, opisana je izrada demo projekta kao i postupak testiranja odnosno mjerenja performansi.

Ključne riječi

Udaljeno upravljanje; RPI; Ulazni uređaji; uinput bibloteka; rawinput biblioteka

Abstract

Nowadays, modern man is surrounded by a large number of different devices, technical and operating systems. Most such systems are designed in a way that at some point they require user interaction. To create such interaction, special devices were designed, which differ from each other depending on their purpose, and which are collectively called input devices. The fact is that in many cases, to manage the system it is necessary to have an input device that must be in close proximity to the system itself, which results in the accumulation of input devices in the user environment. In order to reduce the number of input devices in the user environment, to enable remote control, as well as control in cases when an input device is not present, system for indirect use of input devices are designed. The prototype design of such a system is described through this paper. The stages of software development for the two main components, software application on intermediate device and driver on the target platform are covered. In addition, the development of a demo project is described, as well as the process of testing and measuring performances.

Key words

Remote controll; RPI; Input devices; uninput library, rawinput library

Životopis

Luka Kruljac rođen je 25.3.1997. u Đakovu. Odrastao u Gašincima gdje je i pohađao Osnovnu školu J.A.Ćolnića od 1. do 4. razreda. Osnovnu školu od 5. do 8. razreda pohađa u istoimenoj školi u Satnici Đakovačkoj. Godine 2010. upisuje prirodoslovno-matematičku gimnaziju A.G.Matoš u Đakovu, istu završava 2015. godine kada upisuje program vojnog kadeta, smjer Vojno inženjerstvo. Program kadeta napušta iz osobnih razloga te u listopadu 2015. godine upisuje sveučilišni preddiplomski studij elektrotehnike na Elektrotehničkom fakultet Osijek. Za vrijeme preddiplomskog studija 2016. i 2017. odrađuje prakse u tvrtki Siemens Convergence Creators, a 2018. u tvrtki Inchoo. Nakon završenog preddiplomskog studija komunikacija i informacijskih tehnologija upisuje diplomski studij Automobilskog računarstva i komunikacija na istom fakultetu. Tijekom studija bio je član, a zatim i predsjednik Studentskog Zbora, te član studentskog ogranka IEEE Osijek, gdje također u jednom periodu obnaša neke od dužnosti uprave ogranka. Sudionik je studentskog Work&Travel programa u SAD-u tijekom ljeta 2019., a po za povratku se programa u jesen iste godine počinje raditi kao stipendist u tvrtki RT-RK koju napušta nakon gotovo godinu dana, zatim, od jeseni 2020. godine sudjeluje na Erasmus Internship programu u tvrtki Infineon Technologies u Austriji gdje i danas radi kao software & component validation engineer.

U Osijeku, 26.6.2021.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prilozi

1. Klasni dijagram čitavog rješenja
2. Izvorni kod python skripte za upravljanje kotačima

Elektronička verzija rada

1. Na linku <https://github.com/lkruljac/diplomskiV2> može se pronaći izvorni kod čitavog projekta
2. Na linku https://github.com/lkruljac/diplomskiV2/raw/master/Diplomski%20rad%20-%20Luka%20Kruljac%20\_%20Sep21.docx može se pronaći elektronička verzija ovog dokumenta
3. Na linku https://drive.google.com/drive/folders/19gSh3u\_O7lwJ6STntR2eCAQP0J6EXJQw može se pronaći demo prikazi sustava

