Izrada perifernog uređaja za upravljanje softverom korištenjem Arduino platforme

Poljuha, Matej

Sveučilište u Rijeci – Odjel za informatiku

Jednopredmentni preddiplomski studij informatike

Matej Poljuha

Izrada perifernog uređaja za upravljanje softverom korištenjem Arduino platforme

Završni rad

Mentor: doc. dr. sc. Miran Pobar

Rijeka, 14.7.2020.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 1.6.2020.

Zadatak za završni rad

Pristupnik: Matej Poljuha

Naziv završnog rada: Izrada perifernog uređaja za upravljanje softverom korištenjem Arduino platforme

Naziv završnog rada na eng. jeziku: Development of a peripheral device for software control using the Arduino platform

Sadržaj zadatka:

Izraditi i opisati periferni uređaj koji omogućava interakciju sa softverom na računalu putem odabranih hardverskih kontrola poput tipki, potenciometara ili senzora. Izraditi odgovarajući upravljački program pomoću kojeg korisnik može prilagoditi funkcije izrađenog uređaja za rad s odabranim softverom.

Mentor

doc. dr. sc. Miran Pobar

Voditelj za završne radove

doc. dr. sc. Miran Pobar

Zadatak preuzet: 1.6.2020.

(potpis pristupnika)

Sadržaj

1 Sažetak

Cilj ovog rada bila je izrada perifernog uređaja za upravljanje softverom i pripadajućeg upravljačkog softvera. Prije opisa samog uređaja i softvera opisana je ukratko Arduino platforma, njezine hardverske i softverske komponente te konkretna struktura Arduino Leonardo razvojne pločice koja je korištena u ovom projektu. Nakon opisa Arduino platforme predstavljene su osnovne ideje i koncepti koji su korišteni u izradi i programiranju samog uređaja. Potom su ukratko navedene i objašnjene elektroničke komponente korištene u izradi uređaja. Objašnjeni su i najbitniji dijelovi programskog koda, pogotovo koda koji upravlja Arduinom. Osim Arduino koda objašnjen je i upravljački softver koji se izvodi na računalu i reagira na podatke koje Arduino šalje te izvodi zadane funkcije. Na kraju rada navedena su moguća poboljšanja uređaja i softverskog dijela projekta.

Ključne riječi:

periferni uređaj, serijska komunikacija, Arduino Leonardo, kontakti/terminali, Python, biblioteka, driver softver, GUI softver, konfiguracijska datoteka, simulacija fizičkog pritiska tipkovničke kombinacije

2 Uvod

Kao projekt za demonstraciju rada s Arduino razvojnom pločicom odabran je projekt izrade perifernog uređaja za upravljanje softverom. Ovakvi uređaji su iznimno korisni u raznim okolnostima i često služe specifičnim svrhama koje tradicionalni periferni uređaji poput miša i tipkovnice ne ispunjavaju (ili ispunjavaju na spor/neintuitivan način).

Neki primjeri ovakvih uređaja su:

Slika 1: Tangent Ripple uređaj

- **Tangent Ripple** (Slika 1.) - namijenjen prvenstveno korisnicima softvera za uređivanje fotografija i video snimki. Ovaj uređaj je specijaliziran za brže i efikasnije manevriranje izbornicima za boje, zadatak koji je višestruko sporiji ako se koristi običan miš i/ili tipkovnica

Slika 2: Elgato Stream Deck uređaj

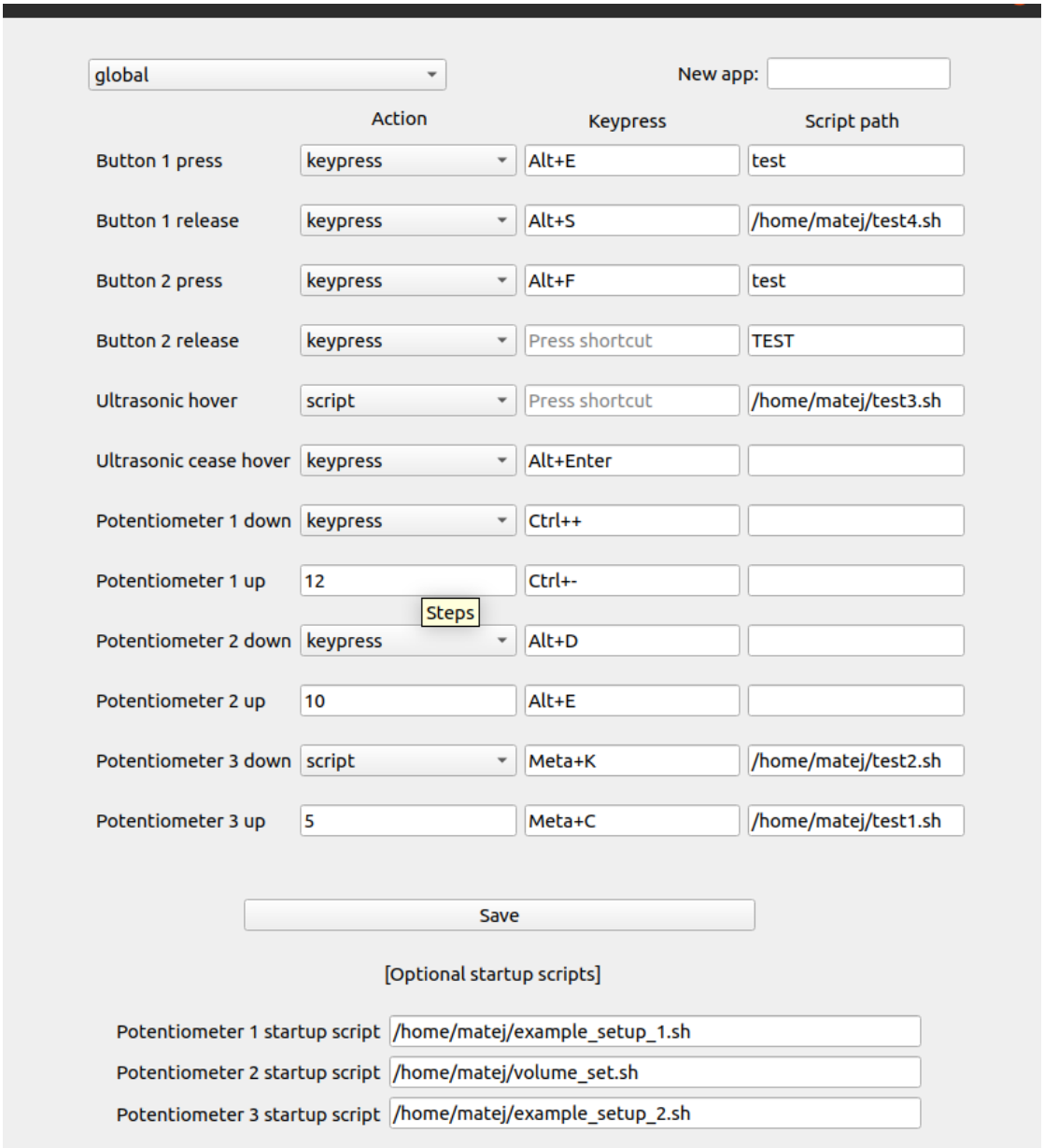
- **Elgato Stream Deck** (Slika 2.) - uređaj namijenjen prvenstveno korisnicima koji uživo prenose sadržaj na platformama poput YouTube.com i Twitch.tv. On omogućuje kreiranje i brz pristup naprednijim funkcijama korisnim za interakciju s publikom ili brzu modifikaciju sadržaja koji se prenosi

Slika 3: Loupedeck+ uređaj

- **Loupedeck+** (Slika 3.) - također namijenjen korisnicima Adobe i Final Cut softvera za uređivanje fotografija i video snimki no različitog pristupa od gorespomenutog Tangent

Ripple uređaja. Ovaj uređaj nije specijaliziran samo za jednu svrhu već ima mnogo ugrađenih funkcija za interakciju s Adobe ili Final Cut softverom.

Svrha ovakvih uređaja je pružiti korisniku brži i/ili intuitivniji način za obavljanje određene funkcije. Neki od njih su programabilni od strane korisnika dok su drugi (poput Loupedeck+ uređaja) pretprogramirani od strane proizvođača za određene svrhe. Ovakvi pretprogramirani uređaji često su fokusirani na manji broj popularnih programa (primjerice Adobe Suite programa) za koje implementiraju razne funkcije koje korisnici često upotrebljavaju. U sklopu ovog projekta izrađeni uređaj je osmišljen da funkcionira na općenitiji način, to jest da nije fokusiran na samo nekoliko programa nego da ima mogućnost rada u većini programa koje korisnik koristi. Fizički, on je implementiran kao niz različitih elektroničkih komponenti i senzora povezanih na jednu Arduino razvojnu pločicu. Konkretno je odabrana pločica Arduino Leonardo zbog jednostavnosti povezivanja svih željenih komponenti i manipulacije te prijenosa rezultantnih podataka na računalo. Ta pločica šalje podatke računalu kad god korisnik interaktira sa spomenutim komponentama na određen način (primjerice rotira potenciometar ili dodirne senzor dodira). Upravljački softver (napisan u programskom jeziku Python) koji je pokrenut na računalu čita te podatke i izvršava određenu akciju zavisno o podatku koji je pročitao, konkretno simulira tipkovnički prečac ili pokreće nekakvu skriptu. Primjer ove funkcionalnosti je situacija u kojoj korisnik može pomicanjem potenciometra promijeniti glasnoću nekog izvora zvuka bilo putem tipkovničkih prečaca ili nekakve skripte. Načinjeni softver se sastoji od 2 skripte i jedne konfiguracijske datoteke u kojoj se čuvaju podatci potrebni za izvođenje funkcija (prethodno spomenute akcije pokretanja skripti i simulacije tipkovničkih prečaca). Jedna skripta je ustvari upravljački softver koji nema grafičko sučelje već ju korisnik jednostavno pokreće u pozadini dok je druga skripta u biti grafičko sučelje za interakciju s konfiguracijskom datotekom (prikazano na Slici 4. i u jednom od sljedećih poglavlja). Ključni elementi ovog softvera, kao i dodatna poboljšanja i optimizacije opisane su u pratećim poglavljima.



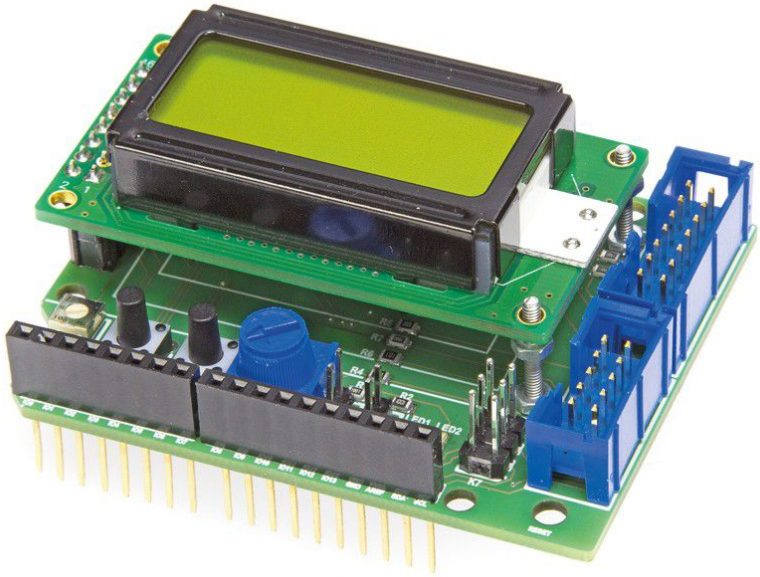
Slika 4: Grafičko sučelje za konfiguraciju

3 Opis Arduino platforme

Arduino je popularan open-source ekosustav koji se sastoji od hardverskih i softverskih komponenata. Razvoj ove platforme je započeo u ranim 2000.-ima na Ivrea Interaction Design institutu u Italiji. Prva Arduino razvojna pločica predstavljena je 2005. godine i originalna svrha joj je bila omogućiti studentima bez iskustva u radu s elektronikom izradu jednostavnih prototipova [ard(a)]. Od tada, tvrtka Arduino uz pomoć sada već znatno veće zajednice proširila je osnovni koncept svog projekta te on sada uključuje širok raspon razvojnih pločica, dodatnih komponenata i softverskih biblioteka. Danas je Arduino platforma često korištena u svrhu brzog prototipiranja, ne samo za studentske i hobi projekte već i u razne komercijalne i znanstvene svrhe.

3.1 Hardverske komponente Arduino ekosustava

Arduino hardverski ekosustav sastoji se većinom od razvojnih ploča različitih specifikacija i namjena. Ove ploče su uglavnom bazirane na Atmelovim mikrokontrolerima AVR® RISC ili ARM arhitekture. Različite modele Arduino ploča razlikuje broj priključaka (kontakata), količine flash i SRAM memorije te fizičke dimenzije i konstrukcija. Osim razvojnih ploča, Arduino ekosustav uključuje i razne druge komponente poput različitih senzora i takozvanih input-output štitova (čiji je primjer prikazan Slikom 5.). Input-output štitovi (eng. *IO shields*) su ploče koje je vrlo jednostavno priključiti na Arduino ploču i proširuju njezinu funkcionalnost, primjerice dodavanjem LCD touch ekrana ili ploče s više ugrađenih senzora.



Slika 5: Arduino LCD štit

3.1.1 Licenciranje

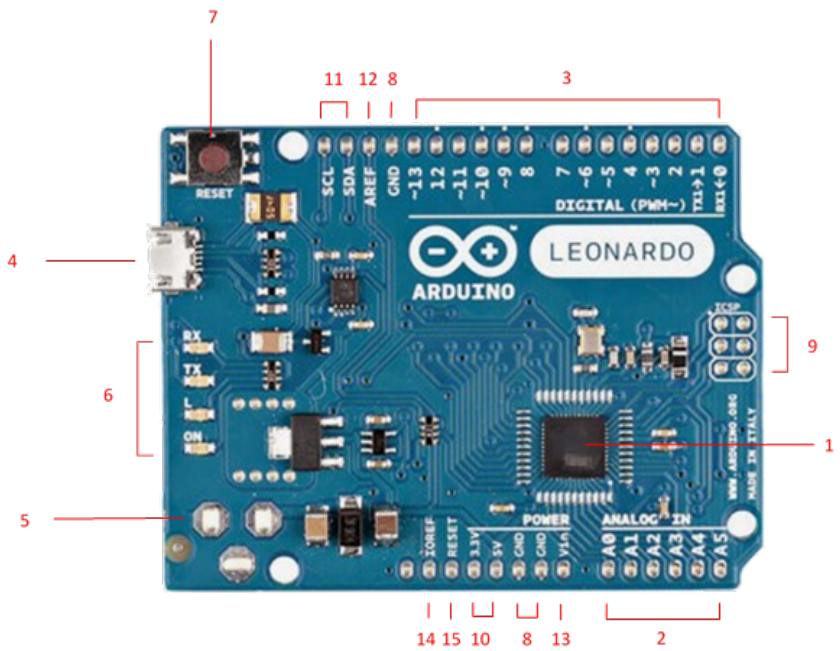
Arduino hardver je open-source. Tvrtka Arduino javno objavljuje nacrte svojih razvojnih pločica u Eagle CAD formatu. Ti nacrti su licencirani po Creative Commons Attribution Share-Alike licenci. Ova licenca dozvoljava osobnu i komercijalnu upotrebu materijala, uz uvjet da izvedeni proizvodi budu licencirani pod istom licencom i da je tvrtka Arduino navedena u zaslugama. Službene Arduino razvojne pločice (jedine koje nose zaštićeno ime Arduino) su proizvod bliske suradnje određenih proizvođača s Arduino razvojnim timom kroz koju je osigurana kvaliteta i kompatibilnost s ostatkom Arduino ekosustava. Neslužbene razvojne pločice proizvedene prema spomenutim open-source nacrtima su također dostupne, ali u svom nazivu ne smiju koristiti ime Arduino već samo Arduino compatible ili neki slični izraz [ard(a)].

3.1.2 Osnovni princip rada

Osnovna ideja Arduino hardverske platforme je da su Arduino razvojne pločice sposobne čitati ulazne signale (poput pritiska gumba ili okretanja potenciometra) i pretvoriti ih u izlazne signale (korisne primjerice za paljenje LED svjetla ili komunikacije s bazom podataka). Ova funkcionalnost ostvaruje se slanjem seta instrukcija mikrokontroleru na ploči [ard(a)]. Postoji relativno velik broj različitih modela Arduino razvojnih pločica, neke su specijalizirane za određene svrhe dok su neke, poput popularnog modela Uno, prikladne za razne projekte. Jedan primjer specijaliziranije pločice je Arduino Mega 2560 čija je definirajuća karakteristika činjenica da ima 54 digitalna terminala (naspram standardnih 14 na standardnoj Uno pločici). Upravo zbog ovakvih karakteristika su Arduino pločice iznimno korisne. One dozvoljavaju čak i ljudima bez prethodnog iskustva u radu s mikrokontrolerima pristup njihovim funkcionalnostima. Arduino pločica se u stvari sastoji od mikrokontrolera i komponenata koje omogućavaju jednostavno povezivanje raznih drugih komponenti i senzora na mikrokontroler bez da korisnik mora sam dizajnirati cijelu ploču koja pruža osnovne komponente za komunikaciju s mikrokontrolerom. Prvenstveno se ovdje misli na razne kontakte, za napajanje, uzemljenje, clock i slično. Iz ovog razloga sam u ovom projektu koristio Arduino pločicu modela Leonardo. Ovaj model vrlo je sličan često korištenom modelu Uno, no uz jednu bitnu razliku, nema odvojeni mikrokontroler za USB komunikaciju već je ta mogućnost implementirana kao softverski dio ATmega32u4 mikrokontrolera. Ova funkcionalnost omogućava računalu da Arduino detektira kao generički USB uređaj što znači da možemo programirati Arduino da direktno simulira pritiske tipki na tipkovnici ili pomiče kursor miša. Ova funkcionalnost je bila razmatrana u ranijim fazama izrade ovog projekta no na kraju nije korištena ali Leonardo pločica je ionako bila prikladna za ovakav projekt pa je stoga ipak upotrijebljena.

3.1.3 Struktura razvojne ploče Arduino Leonardo

U ovom poglavlju opisane su osnovne komponente Arduino razvojnih ploča na primjeru ploče modela Leonardo (Slika 6.) zato jer je konkretno ovaj model ploče korišten u praktičnom dijelu ovog rada. Posebnost ovog modela je to što mikrokontroler ima ugrađenu sposobnost USB komunikacije, bez korištenja odvojenog mikroprocesora specifično za tu svrhu. Ova sposobnost omogućuje računalu da prepozna ploču kao miš i/ili tipkovnicu. [Ard()]



Slika 6: Arduino Leonardo razvojna pločica

1. ATmega32u4 mikrokontroler - 8-bitni mikrokontroler tvrtke Atmel (od 2016. godine u vlasništvu tvrtke Microchip Technology), baziran na AVR® RISC arhitekturi, sadrži 32KB programabilne flash memorije, 2.5KB SRAM-a, 1KB EEPROM-a te USB funkcionalnost (USB 2.0 brzine)

2. Analog ulazi - 6 terminala, A0 do A5, omogućuju očitanja analognih signala

3. Digitalni ulazi - 14 terminala, D0 do D13, terminali s oznakom "~" imaju sposobnost korištenja Pulse Width Modulation (PWM) tehnike za dobivanje analognih rezultata digitalnim putem, PWM radi na način da promjenom vremena koje signal provede primjerice na 0V (fully off) odnosno vremena koje provede na 5V (eng. *HIGH*) simuliramo analogni signal (u ovom slučaju voltažu između 0 i 5 volta)

4. Micro USB konektor - pomoću ovog konektora spajamo ploču s računalom i uploadamo naš program, također služi za napajanje i serijsku komunikaciju

5. Vanjsko napajanje - na slici je prikazano mjesto gdje je moguće dodati priključak za vanjsko napajanje kojim se može napajati Arduino putem AC-DC adaptera ili baterije, preporučeni ulazni napon je 7-12V dok su granice 6-20V

6. LED svjetla - RX (eng. *Receive*) i TX (eng. *Transfer*) svjetla indiciraju slanje i primanje podataka, L svjetlo je ugrađeno LED svjetlo koje je moguće koristiti u programima, ON svjetlo je statusno svjetlo koje indicira da ploča radi

7. Reset gumb - gumb koji ugasi i ponovno upali mikrokontroler i ponovno pokreće program koji je trenutno u memoriji

8. Uzemljenje (eng. *Ground*) - terminali za uzemljenje spojenih komponenata

9. ICSP (eng. *In-Circuit Serial Programming*) - terminali koji služe kao alternativni način za uploadanje programa na ploču, mogu služiti i za popravak oštećenog ili nepostojećeg bootloader programa

10. 3.3V i 5V terminali - omogućuju napajanje vanjskih komponenata odgovarajućim naponom

11. SCL (eng. *Serial Clock*) i SDA (eng. *Serial Data*) - terminali korišteni za I^2C komunikaciju, omogućuju komunikaciju uređaja po master-slave principu

12. AREF (eng. *Analogue REFerence*) - terminal kojim Arduinu dajemo određenu voltažu iz vanjskog izvora da bi ju mogli koristiti kao referentnu voltažu za poboljšanje preciznosti očitanja analognih signala [the AREF pin(2020)]

13. Vin - terminal koji možemo koristiti kao alternativni način napajanja ploče, služi za istu svrhu kao i priključak za vanjsko napajanje

14. IOREF (eng. *Input Output REFerence*) - prikazuje voltažu pri kojoj mikrokontroler operira, korisno pri korištenju input-output štitova

15. RESET terminal - ugasi i ponovno upali mikrokontroler i resetira program koji je trenutno u memoriji, ista funkcionalnost kao i reset gumb

3.2 Softverske komponente Arduino ekosustava

Softverski dio Arduino platforme sastoji se od Arduino IDE-a i raznih softverskih biblioteka za rad s hardverskim komponentama.

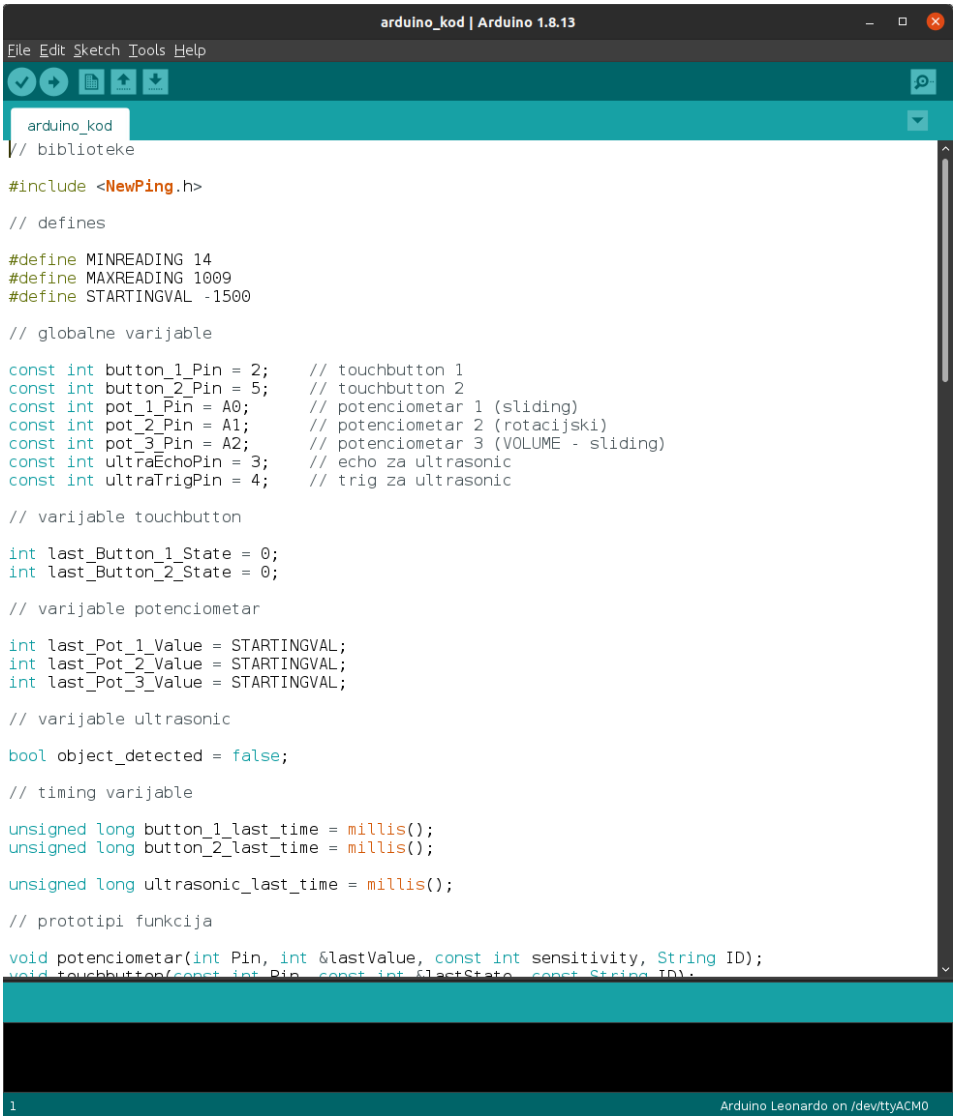
3.2.1 Licenciranje

Kao i hardver, Arduino softver je također open-source. Arduino IDE je licenciran pod GPL licencom (eng. *GNU General Public License*) dok su osnovne biblioteke licencirane pod LGPL licencom (eng. *GNU Lesser General Public License*). Ovo znači da svaka modifikacija osnovnim bibliotekama mora također biti objavljena pod LGPL licencom. Ako su Arduino osnovne biblioteke korištene kao dio firmware-a nekog komercijalnog proizvoda, nije potrebno objaviti izvorni kod cijelog firmware-a već samo objektne datoteke koje omogućavaju ponovno povezivanje tog firmware-a s najažurnijim verzijama osnovnih Arduino biblioteka.[ard(a)]

3.2.2 Arduino programski jezik

Arduino programski jezik baziran je na open-source Wiring programskom okviru za mikrokontrolere. On je u biti set C i C++ funkcija koje je moguće pozvati u kodu. Za kompilaciju koda koristi se avr-g++ compiler što znači da je u kodu dozvoljeno koristiti sve standardne C i C++ konstrukte koji su podržani u tom compiler-u.[ard(a)]

3.2.3 Arduino IDE



Slika 7: Sučelje Arduino IDE

Arduino IDE (čije je sučelje prikazano Slikom 7.) je integrirano razvojno okruženje je alat koji omogućuje komunikaciju i upload programa na Arduino hardver. Postoje dvije varijacije ovog razvojnog okruženja, desktop i web aplikacija. Glavna razlika između ovih varijacija je ta što web aplikacija ne podržava rad s 3rd party hardver-om dok desktop aplikacija to dopušta.

Programi napisani korištenjem Arduino softvera nazivaju se sketch-evi i ekstenzija datoteke je .ino. Jedna od glavnih prednosti ovog okruženja je da podržava više različitih arhitektura i razvojnih pločica, bez njegove funkcionalnosti bi programer morao ručno prenositi programski kod na ploču koristeći hardver specijaliziran za tu pojedinu arhitekturu.[ard(b)]

Glavne mogućnosti Arduino IDE-a:

- uređivanje teksta (koda) s dodatnim funkcionalnostima (primjerice automatsko formatiranje i indentacija)
- prikaz grešaka pri kompilaciji koda
- prikaz zauzeća memorije za pojedine varijable i ukupan kod u konzoli
- odabir specifičnog modela Arduino razvojne ploče
- upload programskog koda u flash memoriju Arduino razvojne ploče
- modifikacija ili uploadanje novog bootloader programa na mikrokontroler pojedine Arduino ploče
- pregled izmijenjenih podataka s povezanim uređajima putem Serial Monitor prozora
- podrška za dodavanje 3rd party hardver-a i softver-a

3.2.4 Biblioteke

Biblioteke pružaju dodatne mogućnosti koje korisnik može implementirati u kodu (eng. *sketch*). U kod se dodaju #include naredbom na vrhu sketch-a. Pri pisanju programa bitno je voditi računa da ove biblioteke također zauzimaju određenu količinu memorije kad se upload-aju na razvojnu ploču.

Neki primjeri:

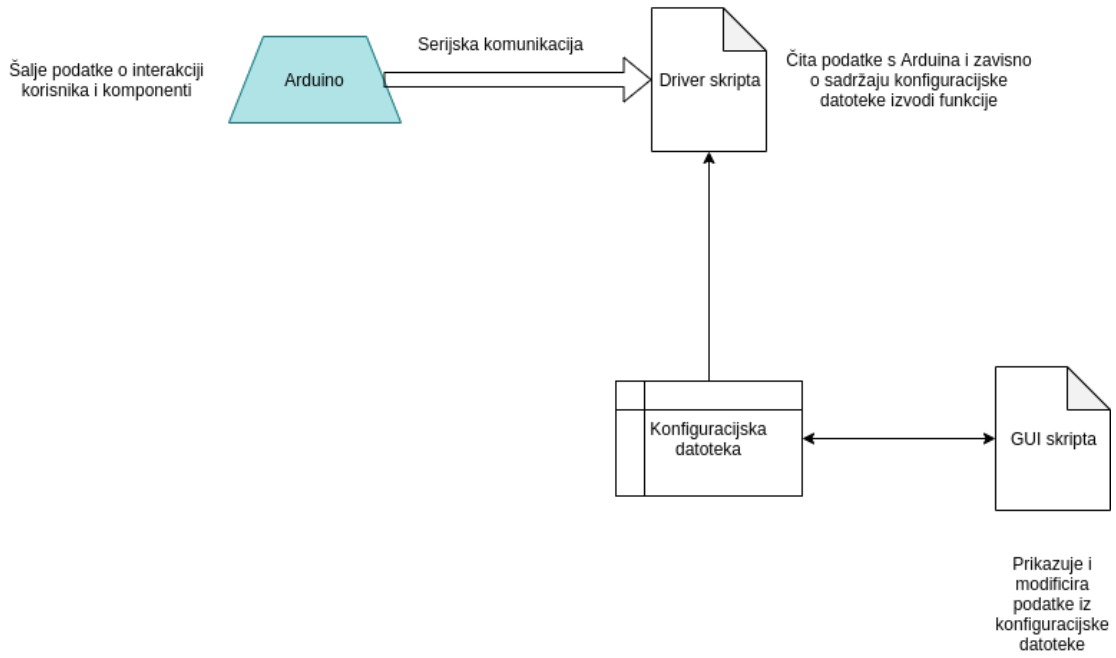
- Mouse - omogućuje kontrolu kursora priključenog računala, primjerice upravljačkom palicom (eng. *joystick*)

- Keyboard - omogućuje simulaciju pritiska određene tipke na tipkovnici, primjerice simulacija jednog pritiska tipke “Enter” ako korisnik zaplješće rukama (korištenjem odgovarajućeg senzora)
- Servo - za kontroliranje spojenih servo motora
- Stepper - za kontroliranje stepper motora
- WiFi - dodaje mogućnost spajanja Arduino ploče na bežičnu WiFi mrežu korištenjem Arduino WiFi štita

4 Periferni uređaj za upravljanje softverom

4.1 Osnovni koncept

Uređaj konstruiran u sklopu ovog rada je primjer perifernog uređaja koji je programabilan od strane korisnika. Njegov koncept rada je sljedeći: razvojna pločica Arduino Leonardo je centralni dio uređaja i ona prima ulazne podatke s komponenti kojima korisnik manipulira, zatim te podatke pretvara u određene linije teksta koje šalje računalu putem micro-USB kabela (koji ujedno služi za napajanje same pločice), program na računalu zatim čita dobivene podatke liniju po liniju te zavisno o pročitanim podacima izvodi određenu funkciju (simulira pritisak tipki na tipkovnici ili izvodi shell skriptu). Korisnik mijenja funkcionalnost svake pojedine komponente spojene na Arduino putem jednostavnog grafičkog sučelja. Slika 8. prikazuje dijagram koji opisuje općenitu ideju rada uređaja.



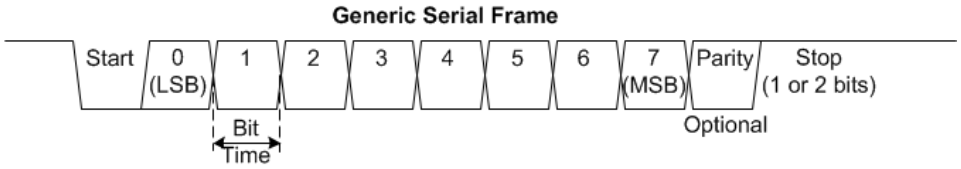
Slika 8: Dijagram osnovnog koncepta rada perifernog uređaja

Program koji čita podatke dobivene od Arduina napisan je u programskom jeziku Python, dok

je grafičko sučelje za konfiguraciju napisanu također u Python-u ali uz korištenje Qt programskog okvira. Sami konfiguracijski podaci spremljeni su u json datoteci.

4.2 Serijska komunikacija

Ključni dio ovog projekta je serijska komunikacija između računala i Arduino Leonardo pločice. Ona se koristi za prijenos podataka o korisničkoj interakciji s uređajem na računalo. Pri serijskoj komunikaciji, kao što joj samo ime govori, podaci se šalju serijski to jest jedan po jedan (bit). Ti podaci su organizirani u takozvane okvire (generalni primjer na Slici 9.), grupe podatkovnih, upravljačkih i kontrolnih bitova organizirane na specifičan način koji mora biti poznat drugom sudioniku u serijskoj komunikaciji.



Slika 9: Okvir u serijskoj komunikaciji

Hardver koji omogućuje ovu komunikaciju na Arduinu je ATmega32u4 mikrokontroler koji ima ugrađen UART (eng. *Universal Asynchronous Receiver-Transmitter*), hardversku komponentu koja omogućava asinkronu serijsku komunikaciju. Asinkrona komunikacija znači da nije potreban *clock* signal kao u sinkronoj komunikaciji već je potreban samo dogovor pošiljatelja i primatelja oko [**?**]:

- brzine slanja i čitanja podataka (eng. *baudrate*)
- razine napona koji predstavlja logički 0 i 1
- značenje višeg i nižeg napona, to jest koji od njih predstavlja logički 0, a koji predstavlja logički 1

U hardverskom pogledu ovakva komunikacija podrazumijeva tri fizičke konekcije (veze):

- zajedničko uzemljenje, da oba sudionika u komunikaciji imaju referentnu točku za očitanje napona
- konekciju za slanje podataka, TX (eng. *transmit*)
- konekciju za primanje podataka, RX (eng. *receive*)

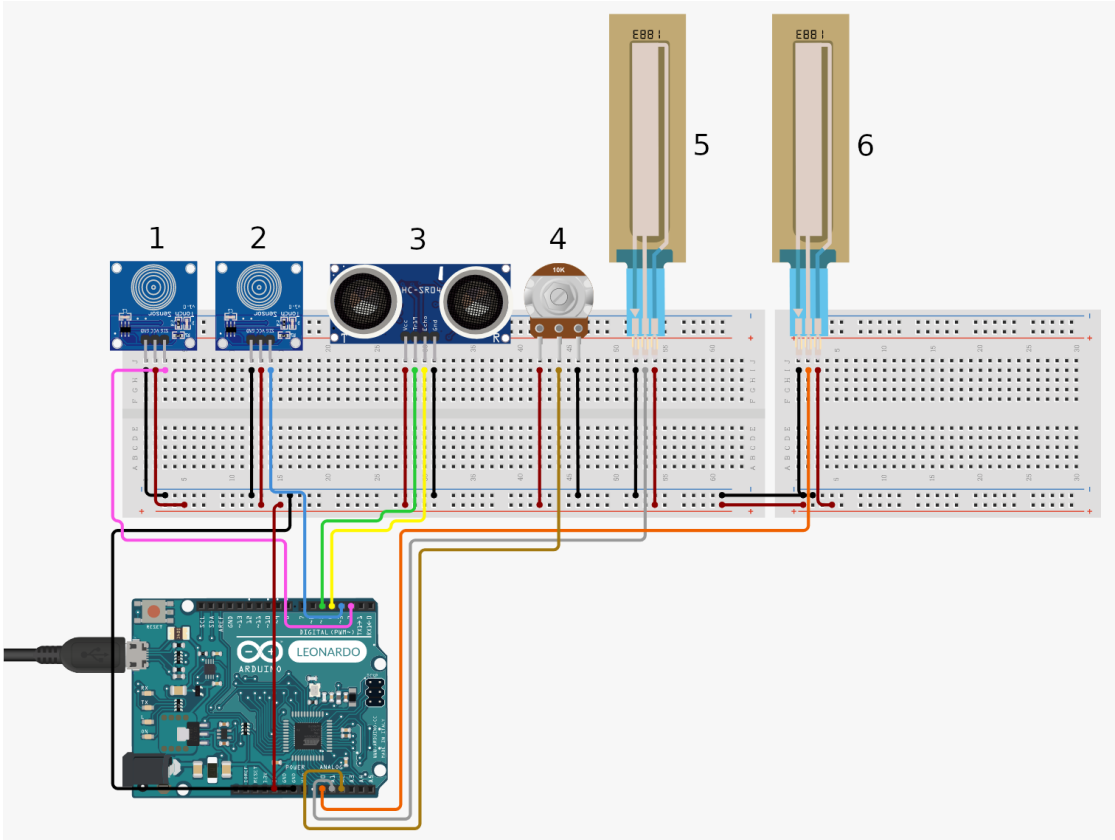
Jedna specifičnost Leonardo modela razvojne pločice je ta što je serijski port virtualan (izveden u softveru) umjesto da je implementiran posebnim mikrokontrolerom odvojenim od glavnog. Sam ATmega32u4 mikrokontroler ima dva načina serijske komunikacije, korištenjem digitalnih kontakata (terminala) 0 (RX) i 1 (TX) za standardnu UART komunikaciju te CDC (eng. *Communications Device Class*) komunikaciju putem USB protokola [Ard()]. UART komunikaciju provodimo korištenjem klase Serial1, dok za USB komunikaciju koristimo klasu Serial. U daljnjim poglavljima bit će spomenuto čitanje serijskih podataka "liniju po liniju", to se ne odnosi na neku specifičnu funkcionalnost serijskog protokola već na činjenicu da su informacije koje Arduino šalje, primjerice informacija da je korisnik dodirnuo senzor dodira, poslane u obliku niza znakova terminiranog znakom za novi red (to jest liniju).

4.3 Korištene komponente

U izradi uređaja korištena je razvojna pločica Arduino Leonardo, ultrasonični senzor udaljenosti, dva kapacitivna senzora dodira te tri potenciometra (dva klizna i jedan rotirajući). Ovi senzori odabrani su radi demonstracije različitih funkcionalnosti ovakvih uređaja no moguće je koristiti mnogo različitih senzora i komponenata kod izrade ovakvih uređaja.

4.3.1 Shema

Ova shema (Slika 10.) prikazuje konfiguraciju uređaja, ilustrativne je prirode no prikazuje način na koji su korištene komponente spojene na Arduino pločicu. Svaka komponenta spojena je na GND i 5V terminale (radi napajanja i uzemljenja) te na odgovarajuće analogne ili digitalne terminale (radi očitanja rezultantnih vrijednosti). Potenciometri su spojeni na analogne terminale jer vraćaju analogne vrijednosti napona dok su kapacitivni senzori dodira i ultrasonični senzor udaljenosti spojeni na digitalne terminale jer vraćaju digitalne vrijednosti (napone 0 ili 5 V, to jest digitalno '0' ili '1').

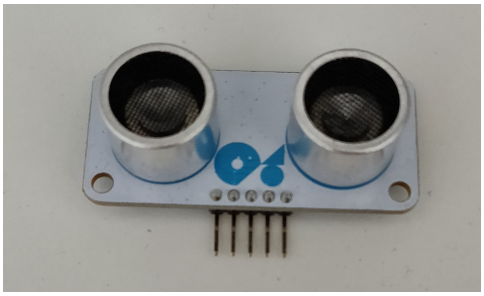


Slika 10: Shema konfiguracije uređaja

Komponente:

1. Kapacitivni senzor dodira
2. Kapacitivni senzor dodira
3. Ultrasonični senzor udaljenosti
4. Rotirajući potenciometar
5. Klizni potenciometar
6. Klizni potenciometar

4.3.2 Ultrasonični senzor udaljenosti



Slika 11: Korišteni ultrasonični senzor udaljenosti

Ovaj senzor koristi se za mjerenje udaljenosti pomoću ultrazvučnih valova. Sastoji se od odašiljača i prijamnika te 4 kontakta: VCC (napajanje), GND (uzemljenje), TRIG i ECHO. Kada kontakt TRIG postavimo na HIGH (pružimo napon od primjerice 5V) odašiljač proizvodi i šalje zvučni val vrlo visoke frekvencije u određenom smjeru, ako se taj zvučni val sudari i odbije od nekog predmeta (unutar efektivnog dometa senzora), prijamnik registrira taj odbijeni val. Sklopovlje na senzoru potom postavlja ECHO kontakt na vrijednost HIGH onu količinu vremena koliko je bilo potrebno zvučnom valu da bude registriran od strane prijamnika. Na osnovu vremena koje ECHO kontakt provede na HIGH vrijednosti možemo jednostavnom formulom dobiti udaljenost od predmeta.

Ta formula je:

$$D = c * \frac{t}{2}$$

D - udaljenost

c - brzina zvuka u zraku temperature 20°, iznosi 343 m/s

t - vrijeme između odašiljanja vala i primitka na prijamniku (vrijeme dijelimo s 2 iz razloga što je to ukupno vrijeme koje uključuje i vrijeme potrebno da val dođe do objekta i vrijeme da se vrati do prijamnika)

U izradi uređaja za projekt korišten je ultrasonični senzor HC-SR05 tvrtke Velleman (prikazan na Slici 11.)[vel(b)]. Specifikacije ovog senzora su sljedeće:

- napon iznosi 4.5 do 5.5 volta
- potrebna struja je 10-40 mA
- proizvodi ultrasonične valove frekvencije 40 KHz
- kut mjerenja je 15°
- preciznost dobivene udaljenosti je 0.3 cm
- detektira udaljenosti od 2 cm - 4.5 m
- Senzor se aktivira i odašilje zvučni val kad je TRIG (trigger) kontakt postavljen na HIGH vrijednost (pušten napon od 5V, digitalno 'ON' to jest '1') minimalno 10 μS.

U ovom projektu ovaj senzor je korišten za detekciju pokreta, na način da šalje signal ukoliko korisnik stavi dlan (ili bilo koji objekt) ispred senzora, ispod određene udaljenosti. Također odašilje se i drukčiji signal nakon što korisnik ukloni objekt nakon njegove detekcije. Ovaj senzor nije idealan za ovakvu svrhu, originalna namjera bila je koristiti infracrveni senzor pokreta koji bi bio primjereniji za detekciju pokreta no konkretni senzor koji sam posjedovao je bio relativno nekvalitetan i nije imao adekvatnu funkcionalnost (bio je spor, preosjetljiv te mu je trebalo dulje vrijeme za početak rada).

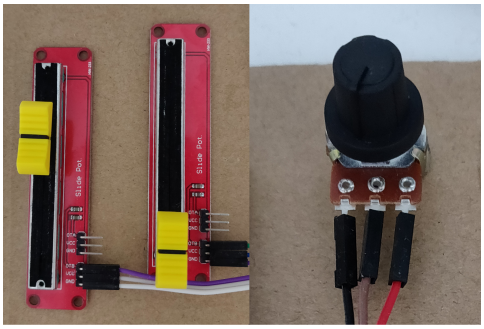
4.3.3 Kapacitivni senzor dodira

Slika 12: Korišteni kapacitivni senzori dodira

Ovaj senzor koristi se kao prekidač to jest gumb. Sadrži 3 kontakta: VCC, GND i S (signal). Sam senzor se sastoji od dva vodiča odvojenih izolatorom te sklopovlja. Funkcionira na način da sklopovlje pri određenim intervalima mjeri trenutni kapacitet. Ukoliko korisnik dotakne senzor, taj kapacitet se efektivno pribraja kapacitetu nakon čega senzor detektira promjenu i postavlja signal kontakt na HIGH vrijednost.

U izradi uređaja za projekt korišten je kapacitivni senzor dodira VMA305 tvrtke Velleman (prikazan na Slici 12.)[vel(a)].

4.3.4 Potenciometar



Slika 13: Korišteni potenciometri, lijevo klizni, desno rotacijski

Potenciometar ili promjenljivi električni otpornik je jednostavna elektronička komponenta nešto kompleksnija od običnog otpornika. On se sastoji od dva statična terminala povezana otpornikom u obliku trake i jednog pomičnog terminala (klizača). Princip rada je da klizač dijeli ukupni otpor potenciometra Rna otpore r i $R - r$. Trošilo je spojeno na otpor r. U ovom slučaju trošilo je Arduino pločica koje na analognom terminalu očitava vrijednosti napona dobivenog pri otporu r (očitane vrijednosti su u rasponu 0 - 5V). [Malesevic(2018)]

U projektu korištene su dvije vrste potenciometara, dva klizna potenciometra te jedan rotacijski (prikazani na Slici 13.). Klizni potenciometri su vrijednosti 10 kΩ dok je rotacijski vrijednosti 100 kΩ. Obje vrste imaju tri terminala: GND, VCC i signal. Terminal signal spojen je na analogni terminal Arduino pločice.

4.4 Ugrađeni kod (Arduino kod)

U ovom poglavlju ću ukratko opisati bitnije dijelove Arduino koda.

4.4.1 Inicijalizacija globalnih varijabli

U sljedećem kodu inicijaliziraju se globalne varijable korištene u radu programa.

```
// globalne varijable
const int button_1_Pin = 2;     // touchbutton 1
const int button_2_Pin = 5;     // touchbutton 2
const int pot_1_Pin = A0;       // potenciometar 1 (sliding)
...
// varijable touchbutton
int last_Button_1_State = 0;
...
// varijable potenciometar
int last_Pot_1_Value = STARTINGVAL; //
...
// timing varijable
unsigned long button_1_last_time = millis();
...
```

Ove varijable zauzimaju dio memorije na Arduinu te je bitno voditi računa o njihovoj uporabi. One se koriste isključivo ako neke podatke trebamo sačuvati kroz više iteracija loop() funkcije. Timing varijable su varijable koje se koriste za optimizaciju. Njihov mehanizam bit će prikazan u poglavlju o loop() funkciji.

4.4.2 Setup()

Setup() funkcija je standardna funkcija koja se izvodi pri paljenju pločice.

```
// setup funkcija
void setup() {
  pinMode(button_1_Pin, INPUT);
  ...
  Serial.begin(115200);
  while(!Serial);
}
```

U ovoj funkciji konfiguriraju se pojedini kontakti da se ponašaju kao ulazni elementi. Ovaj postupak stavlja te kontakte u takozvano stanje visokog otpora što pojednostavljuje implementaciju komponenata poput kapacitivnih senzora dodira. Serial.begin() funkcija otvara serijsku komunikaciju uz parametar čije je značenje broj prenesenih bitova u sekundi - takozvani *baudrate* (ovdje konkretno postavljeno na 115200 bitova po sekundi). Naredba "while(!Serial)" je specifična za određene modele Arduino razvojnih pločica. Većina Arduino pločica, konkretno one s odvojenim USB->Serial mikročipom, se ponovno pokrene nakon što se otvori serijski port (funkcijom Serial.begin()) dok Leonardo pločica nema takav mikročip te se stoga ne resetira pri otvaranju serijskog porta. Stoga program, zbog gorespomenute linije koda, čeka dok se serijska komunikacija ne uspostavi te tek onda kreće u izvođenje glavne petlje.

4.4.3 Loop()

Loop funkcija je glavna funkcija Arduino programa. Ona se krene izvoditi nakon što se završi izvođenje setup() funkcije te se izvodi dok god se pločica ne ugasi.

```
1  void loop() {
2    // POTENCIOMETRI
3      potenciometar(pot_1_Pin, last_Pot_1_Value, 5, "POT_1:");
4      ...
5    // TOUCHBUTTONI
6    if(millis() - button_1_last_time >= 30)
7    {
8      button_1_last_time = millis();
9      touchbutton(button_1_Pin, last_Button_1_State, "Button_1");
10   }
11     ...
12   // ULTRASONIC
13   if(millis() - ultrasonic_last_time >= 51)
14   {
15     ultrasonic_last_time = millis();
16     ultrasonic(12, object_detected);
17   }
18 }
```

U ovoj funkciji pozivaju se funkcije koje očitavaju vrijednosti pojedinih senzora ili komponenata. Funkcije koje očitavaju potenciometre se izvode pri svakom izvođenju loop() funkcije iz razloga što je stanje potenciometara potrebno provjeravati što je brže moguće. Stanje ostalih komponenata se provjerava periodično, korištenjem uvjeta koji provjeravaju je li prošao određen broj milisekundi od zadnjeg izvođenja odgovarajućeg bloka naredbi. U ovu svrhu koriste se globalne *timing* varijable (stanje senzora dodira se provjerava svakih 30 milisekundi dok se stanje ultrasoničnog senzora udaljenosti provjerava svakih 50 milisekundi). Alternativa ovakvom

izvođenju je korištenje prekida (eng. *interrupts*). Prekidi su korisni ako ne trebamo konstantno provjeravati stanje nekog senzora, već želimo samo dobiti informaciju ako se njegovo stanje promijeni. Ovakva implementacija bi bila najkorisnija primjerice za provjeru stanja potenciometara no ovdje je korišten prethodno spomenut način s globalnim *timing* varijablama iz razloga što je implementacija prekidnih signala na Arduino Leonardo pločici moguća ako je komponenta spojena na digitalne ulaze, a potenciometri nažalost nisu. Iako u teoriji moguća je i implementacija na analognim terminalima no ona zahtijeva ručno postavljanje registara u analog-digital konverteru što ima potencijal uvesti dodatne komplikacije pa je stoga spomenuti način [ADC()].

4.4.4 Funkcije očitanja senzora

Funkcija za očitanje kapacitivnih senzora dodira

Ova funkcija, zavisno o tome je li korisnik dodirnuo ili prestao dodirivati senzor dodira šalje serijskom komunikacijom odgovarajuće poruke. Parametri funkcije su oznaka kontakta (*Pin*), globalnu varijablu zadnjeg stanja senzora (*lastState*) te string koji služi za razlikovanje senzora pri čitanju serijskih poruka (*ID*).

```
1  void touchbutton(int Pin, int &lastState, String ID)
2  {
3    int currentState = digitalRead(Pin);
4    if (currentState != lastState)
5    {
6      if (currentState == HIGH)
7      {
8        Serial.print(ID);
9        Serial.println("_on");
10     }
11     else
12     {
13       Serial.print(ID);
14       Serial.println("_off");
15     }
16     lastState = currentState;
17   }
18 }
```

Funkcija za očitanje vrijednosti potenciometara

Argumenti funkcije služe istim svrhama kao i argumenti gorespomenute funkcije za očitanje senzora dodira, uz iznimku parametra *sensitivity* koji se koristi radi smanjenja broja operacija i radi stabilnosti očitanja. Zavisno o specifikacijama potenciometra i interferenciji okoline pojavlja se problem da su očitanja potenciometra nestabilna, to jest mijenjaju se čak i kad je potenciometar netaknut. Ovo ponašanje izbjegava se na način da se vrijednost potenciometra šalje na serijski port samo ako se dovoljno promijenilo od zadnjeg poslanog stanja (sačuvanog u globalnoj varijabli). Također, u memoriji se čuvaju stvarna očitanja dobivena analogRead() funkcijom. Ova očitanja su u rasponu od 0 do 1023 iz razloga što Arduino Leonardo koristi 10-bitni ADC (eng. *Analog to Digital Converter*). Ovaj pretvarač je nužan jer potenciometar proizvodi analogne vrijednosti napona (između 0 i 5 V) koje Arduino mora na neki način pretvoriti u digitalne podatke (nule i jedinice). Pošto je spomenuti ADC 10-bitni dio to znači da ima rezoluciju od 2^{10} to jest 1024 bita što efektivno znači da Arduino može detektirati promjenu u naponu koja iznosi $5V/1024 = 4.9mV$ [ADC()]. Ispisuju se samo vrijednosti skalirane na raspon 0-100 radi jednostavnije daljnje obrade na računalu. Također promjene očitanih vrijednosti koje su ispod 14 ili iznad 1009 se ne razmatraju (funkcija *constrain()*). Ovo je učinjeno iz razloga što zbog fizičke konstrukcije korištenih potenciometara nije moguće konzistentno očitativati vrijednosti pri krajevima kontakata. Konkretno, korišteni sliding potenciometri su bolje kvalitete izrade te im je moguće očitati pun raspon vrijednosti napona (položaja) dok je rotacijski potenciometar nešto nestabilniji i nije mu moguće očitati vrijednosti pri vrhu i dnu raspona 0-1023. Ove optimizacije su rađene iz razloga što je nužno da se svaki pomak potenciometra ispravno i konzistentno prenese kao poruka putem serijske komunikacije, no ni one nisu bile dovoljne da u potpunosti garantiraju slanje poruke. U jednom od sljedećih poglavlja (4.5.1.) bit će objašnjen mehanizam koji je implementiran da funkcionalnost uređaja ostaje jednaka čak i ako iz nekog razloga pomak potenciometra ne dođe do upravljačkog programa (ili jer samo nije očitan ili jer nije uopće poslan).

```
1 void potenciometar(int Pin, int &lastValue, const int
      sensitivity, String ID)
2 {
```

```
3    int currentValue = analogRead(Pin);
4    int diff = currentValue - lastValue;
5    int mappedCurrentValue = map(currentValue, MINREADING,
        MAXREADING, 0, 100);
6    mappedCurrentValue = constrain(mappedCurrentValue, 0, 100);
7    int mappedLastValue = map(lastValue, MINREADING, MAXREADING,
        0, 100);
8    mappedLastValue = constrain(mappedLastValue, 0, 100);
9
10   if ((mappedCurrentValue != mappedLastValue) && (abs(diff) >
        sensitivity))
11   {
12     Serial.print(ID);
13     Serial.println(mappedCurrentValue);
14     lastValue = currentValue;
15   }
16   else if (lastValue == STARTINGVAL)
17   {
18     Serial.print(ID);
19     Serial.println(mappedCurrentValue);
20     lastValue = currentValue;
21   }
22 }
```

Funkcija za očitanje ultrasoničnog senzora udaljenosti

Ova funkcija očitava udaljenost objekta od senzora koristeći metodu sonar.ping_cm() iz biblioteke NewPing (autora Tim-a Eckel-a). Ovaj senzor se u ovom projektu koristi kao senzor pokreta te se stoga poruka šalje serijskom komunikacijom jedino ako je objekt detektiran na udaljenosti manjoj od neke zadane vrijednosti (parametar *max_distance,* koji konkretno iznosi 12 cm). Jednakim principom kao i senzori dodira, šalje se i poruka kad korisnik prestane interakciju (u ovom slučaju odmakne objekt nakon što ga senzor detektira). Biblioteka *NewPing* je korištena zbog boljih performansi od standardnih biblioteka. Njezina funkcija *ping_cm()* uzrokuje odašiljanje zvučnih valova do 30 puta u sekundi te vraća preciznije udaljenosti. Rad ove funkcije može se ukratko prikazati na sljedeći način: funkcija postavlja TRIG terminal na vrijednost 0 (LOW) na nekoliko mikrosekundi kako bi osigurala da je početno stanje LOW, nakon čega postavlja vrijednost TRIG terminala na 1 (HIGH) za sljedećih 10 mikrosekundi (točan puls potreban senzoru nakon čijeg očitanja proizvede zvučni val) nakon toga čeka povratni signal nakon kojeg napravi izračun udaljenosti prema formuli opisanoj u jednom od prethodnih poglavlja. Naposljetku vraća dobivenu udaljenost u centimetrima.

```
1  void ultrasonic(const int max_distance, bool &object_detected)
2  {
3    int distance = sonar.ping_cm();
4    if ((distance > 0) && (distance < max_distance) && (
       object_detected == false)){
5      object_detected = true;
6      Serial.println("Ultrasonic_found");
7    } else if ((distance > max_distance) && (object_detected ==
       true))
8    {
9      object_detected = false;
10     Serial.println("Ultrasonic_lost");
11   }
12 }
```

4.5 Kod na računalu

Osim Arduino koda, u sklopu projekta kreirane su i dvije skripte od kojih prva služi za samo izvršavanje funkcija kad korisnik interaktira sa senzorima (nadalje referiran imenom driver) dok drugi dio služi korisniku za postavljanje konfiguracije i dalje se referencira imenom konfiguracijski GUI (popis funkcija koje izvodi pojedina komponenta, spremljen u json datoteci). Obje skripte napisane su u programskom jeziku Python.

4.5.1 Driver dio

Korištene nestandardne biblioteke:

- pyserial - za serijsku komunikaciju
- psutil - za dohvaćanje informacija o trenutno aktivnim procesima
- python-libxdo - za interakcijom s xdotool alatom (alat za interakciju s X11 sustavom za prikaz prostora)
- pyautogui - za simulaciju fizičkih pritisaka tipki na tipkovnici

Svrha ove skripte je da promatra serijski port na koji je povezan Arduino i čita podatke koje Arduino šalje. Nakon što pročita liniju (korištenjem metode readline() iz biblioteke pyserial), primjerice liniju "Button_1_on", ova skripta izvodi ili simuliranu kombinaciju tipki ili pokreće shell skriptu, zavisno koja funkcionalnost je zapisana u konfiguracijskoj datoteci. Također skripta ima sposobnost izvođenja različitih funkcija zavisno o prozoru koji korisnik trenutno ima aktivno, primjerice korisnik može odabrati da pomicanjem potenciometra smanjuje ili povećava glasnoću zvučnika dok je u internetskom pregledniku no da pomicanjem istog potenciometra povećava ili smanjuje zoom dok je u pregledniku slika. Za implementaciju ove funkcionalnosti se koriste biblioteke python-libxdo te psutil kao što je prikazano u sljedećoj funkciji:

```
"""
Funkcija dohvaca ime prozora koji je trenutno fokusiran
"""
def get_current_window():
    try:
        active_window = xdo.get_pid_window(xdo.
            get_active_window())
        active_window = psutil.Process(active_window).name()
        return active_window
    except:
        pass
```

Funkcijom get_pid_window() dohvaća se process ID trenutno prozora koji je trenutno fokusiran te se zatim dohvaća njegovo ime korištenjem name() metode iz klase Process.

Simulacija pritiska tipki na tipkovnici se izvršava pomoću biblioteke pyautogui koja to omogućuje. Funkcija koja se poziva u ovu svrhu je:

```
pyautogui.hotkey(*potentiometer_keypress)
```

Ova funkcija kao argument prima popis tipki, primjerice ['Ctrl', 'P'] te simulira njihov pritisak te otpuštanje suprotnim redoslijedom (efektivno kao da korisnik pritisne tipke CTRL i P te ih otpusti).

Izvođenje shell skripti izvodi se pomoću standardne biblioteke subprocess, konkretno funkcijom

```
subprocess.run()
```

Korisnik jednostavno putem GUI sučelja unese apsolutni put do željene skripte i tu skriptu može izvesti primjerice kada mahne rukom ispred ultrasoničnog senzora ili pomakne potenciometar. Specifična situacija na koju je bilo potrebno obratiti pozornost je situacija u kojoj se pomak potenciometra ne očita na ispravan način, to jest ako korisnik prebrzo pomakne potenciometar, moguće je da upravljački program recimo dobije informaciju da je vrijednost potenciometra u jednom trenutku 55 dok je u sljedećem 62 što bi značilo da se desio pomak od 7 vrijednosti bez

da je program bio svjestan te promjene. Upravo kao rješenje ovakvog problema implementiran je sljedeći dio koda. Kod u principu pri svakom primitku poruke koja sadrži identifikator potenciometra i njegovu novo-očitanu vrijednost provjerava za koliko je ta nova vrijednost različita od zadnje poznate vrijednosti i poziva funkciju ispravan broj puta. Ukoliko ne bi bilo ovog dijela mogla bi nastati situacija u kojoj se preskoči ključno očitanje potenciometra pri kojem se neka funkcionalnost trebala izvesti što bi dovelo do nekonzistentnog i neuporabljivog ponašanja.

```
1 magic_number = 1 if last_values[event] < int(value) else -1
2
3 l = sum(1 for v in range(last_values[event], int(value) +
    magic_number, magic_number) if v != last_values[event] and v
     % data['apps'][active_window][event]['steps'] == 0)
4     for i in range(l):
5         try:
6             # poziv funkcije koja izvodi odgovarajuću akciju
7             potentiometer_action(data=data, active_window=
                active_window, senzor=event, direction=direction
                )
```

4.5.2 Konfiguracijska datoteka

Ova jednostavna tekstualna datoteka sadrži sve akcije koje korisnik želi izvesti kad manipulira pojedinim senzorom. Ona je formata json. Json (JavaScript Object Notation) je tekstualni format kojeg strojevi lako pretražuju i generiraju te je stoga odabran za korištenje u ovom projektu. Struktura ove konfiguracijske datoteke je sljedeća:

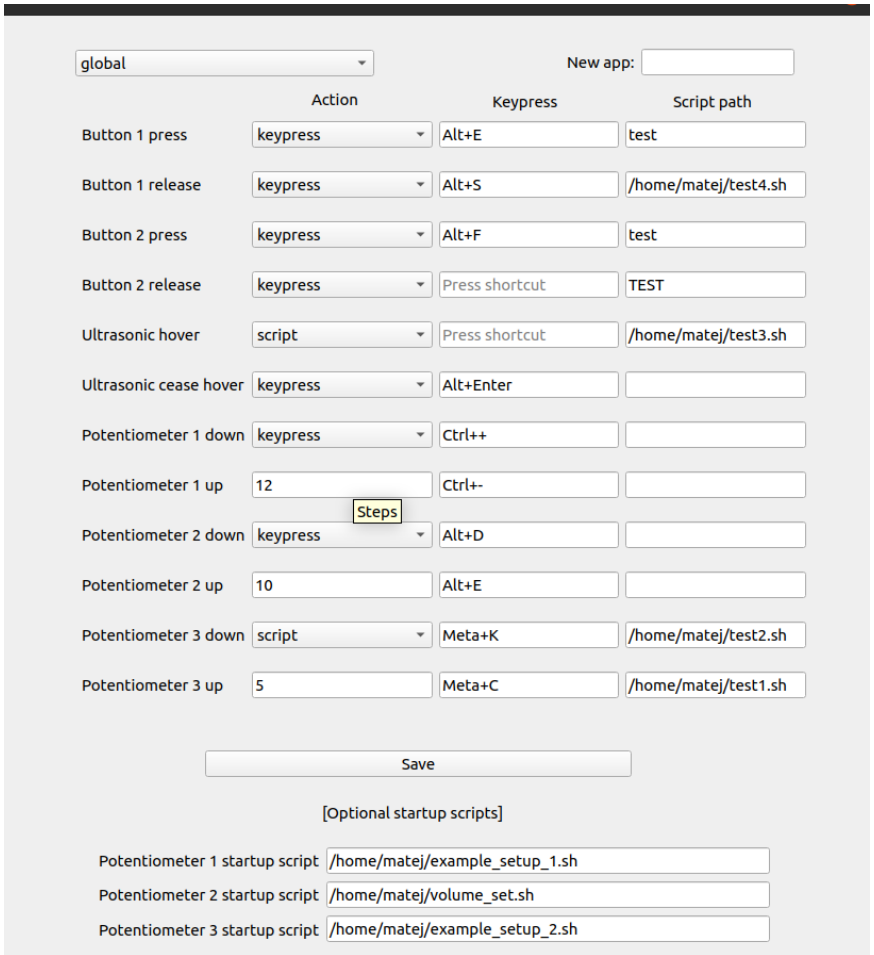
```
1  "setup": {
2          "POT_1": "/home/user/startup_script_1.sh",
3          "POT_2": "",
4          "POT_3": ""
5      },
6      "apps": {
7          "global": {
8              "POT_1": {
9                  "which": "keypress",
10                 "keypress": {
11                     "up": "ctrl+shift+p",
12                     "down": ""
13                 },
14                 "script": {
15                     "up": "",
16                     "down": ""
17                 },
18                 "steps": 10
19             },
20             ...
21             "Button_1_on": {
22                 "which": "script",
23                 "keypress": "",
24                 "script": "/home/user/example_script.sh"
```

```
25                      } ,
26                  . . .
27                  } ,
```

Prva razina se sastoji od ključeva *setup* i *apps*. *Setup* sadrži apsolutne puteve do skripti koje se izvode pri pokretanju programa (ukoliko korisnik želi implementirati neku funkcionalnost koja zahtijeva da računalo zna prvotni položaj potenciometara). Ključevi unutar *apps* razine obilježavaju ime programa koji, ako je fokusiran, mijenja reakcije na interakciju sa senzorima.

4.5.3 Konfiguracijski GUI

Ovaj dio projekta također je napisan u Python programskom jeziku, no uz korištenje Qt programskog okvira. Konkretno korištena je varijanta Qt programskog okvira pod imenom "Qt for Python". Qt for Python je projekt čiji je cilj kreirati potpuni port Qt 5 programskog okvira (originalno napisanog za C++ programski jezik) u jezik Python kreiranjem i održavanjem modula PySide2. Svrha ove skripte je da korisniku omogućuje brzu i jednostavnu konfiguraciju perifernog uređaja.



Slika 14: Grafičko sučelje za konfiguraciju

Struktura ovog grafičkog sučelja vrlo je jednostavna (prikazana je na Slici 14.). Korisnik pomoću padajućeg izbornika bira za koji program želi pregledati/modificirati konfiguraciju (opcija

global je osnovna konfiguracija koja se koristi ako je u fokusu prozor za koji nije stvorena konfiguracija). Korisnik također može stvoriti novu konfiguraciju za pojedini program upisivanjem imena tog programa u polje *New app*. Nakon što korisnik odabere konfiguraciju prikazuju mu se trenutne vrijednosti u poljima *Keypress* i *Script path* (oba polja mogu biti ispunjena u isto vrijeme no "aktivno" je ono koje je odabrano u padajućem izborniku *Action*). Korisnik mijenja ove vrijednosti upisivanjem apsolutne putanje do shell skripte za polje *Script path*, te pritiskom miša na polje *Keypress* nakon čega pritisne željenu kombinaciju tipki na tipkovnici. U tekstualna polja *Steps* upisuju se brojevi koji simboliziraju učestalost izvođenja odabrane funkcionalnosti pri pomicanju potenciometra. Primjerice ukoliko korisnik želi da pomicanjem potenciometra mijenja glasnoću zvučnika (pretpostavimo da se ta glasnoća kreće između 0 i 100) za 1 svaki put kad pomakne potenciometar u željenom smjeru on će u to polje upisati broj 100 (jer želi 100 "koraka"). Ako korisnik želi primjerice povećavati i smanjivati razinu povećanja prikaza (zumiranje), pretpostavimo da je početna razina 100% te da su koraci zumiranja 125%, 150% i 200%, korisnik će u steps polje upisati 4 te će biti potenciometar biti efektivno podijeljen na 4 dijela na način da korisnik mora pomaknuti potenciometar za veću vrijednost kako bi promijenio korak zumiranja. Na kraju korisnik sprema promjene u konfiguracijsku datoteku klikom na gumb *Save*.

4.6 Moguća poboljšanja projekta

U svom trenutnom obliku, uređaj za ostvarivanje osnovne funkcionalnosti zahtijeva relativno velik broj nestandardnih biblioteka te je napravljen primarno za GNU Linux operativne sustave koji koriste X11 sustav prozora.

Puna lista zahtjeva:

- GNU Linux operativni sustav koji koristi X11 sustav prozora
- xdotool alat za automatizaciju
- Python 3.x.x
- nestandardne biblioteke: pyserial, python-libxdo, psutil, pyautogui, PySide2, dpath

Kod ovakvih perifernih uređaja koji simuliraju fizičke pritiske tipki na tipkovnici najveći problem je podrška za sve aktualne platforme (GNU Linux, Windows, MacOS, BSD varijacije...). Vrlo je zahtjevno implementirati jedan alat koji ima potpunu funkcionalnost na svakoj od ovih platformi, ponajviše zbog toga što svaki od ovih sustava ima različit način interpretacije pritiska tipke na tipkovnici. Konkretan primjer, u ranijem stadiju projekta za postavljanje željene kombinacije tipki korištena je biblioteka python-keyboard. Korisniku bi pritisak na *Windows* tipku (na Linux sustavima *Meta* ili *Super* tipka) bio prepoznat kao pritisak tipke *Alt* no samo na pojedinim distribucijama GNU Linux sustava. Također, osim multiplatformskog rješenja za tipkovničke tipke potrebno bi bilo implementirati i posebne sustave za prepoznavanje trenutno aktivnog prozora za svaku pojedinu platformu. Što se tiče konkretnog uređaja kreiranog u sklopu ovog projekta i pripadajućeg koda, moguća su mnoga poboljšanja. Prvo i najočitije poboljšanje je redizajn strukture konfiguracijske datoteke. Ovo poboljšanje bi omogućilo i optimizaciju ostalih skripti koje pristupaju tim podacima. Daljnje poboljšanje bi bilo kombiniranje konfiguracijske i driver skripte u jednu skriptu s boljim grafičkim sučeljem koje bi omogućilo konfiguraciju dodatnih elemenata uređaja.

5 Zaključak

Periferni uređaji za upravljanje softverom vrlo su koristan alat, pogotovo profesionalcima koji često koriste jedne te iste funkcije u nekom alatu, koje su inače skrivene iza nekoliko izbornika i klikova. Izradi ovakvih uređaja može se pristupiti na nekoliko različitih načina no u ovom radu odlučio sam se na pristup u kojem Arduino pločica šalje predodređene kontrolne kodove računalu kad korisnik interaktira s nekim senzorom te potom upravljački program koji se izvodi na računalu reagira na te kodove na način specificiran u konfiguracijskoj datoteci. Naravno pristupi koji ne zahtijevaju konstantnu izvođenje upravljačkog programa ili oni koji su specijalizirani i optimizirani za rad s manjom skupinom programa (često Adobe kolekcija programa) su mnogo praktičniji za korištenje u izradi komercijalnog proizvoda ovog tipa. Svoj pristup sam odabrao iz razloga što prikazuje općenitu funkcionalnost ovakvih uređaja. Kroz ovaj rad objašnjen je odabrani pristup u cijelosti, zajedno s teorijskom podlogom Arduino platforme te elektroničkih komponenti korištenih u izradi uređaja. Na kraju predložena su i mogućnosti poboljšanja ovog projekta te glavni izazovi s kojima se potrebno suočiti u izradi ovakvog uređaja.

6 Popis izvora

6.1 Online izvori

1. https://www.qt.io/qt-for-python
2. https://docs.python.org/3/
3. https://pyautogui.readthedocs.io/en/latest/
4. https://python-libxdo.readthedocs.io/en/latest/
5. https://pythonhosted.org/pyserial/
6. https://psutil.readthedocs.io/en/latest/
7. https://pypi.org/project/dpath/
8. https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home#!constructor

6.2 Izvori slika

1. https://www.elektor.com/arduino-experimenting-shield-2-0-module-160593-91
2. https://store.arduino.cc/arduino-leonardo-without-headers
3. https://www.tangentwave.co.uk/product/ripple/
4. https://www.elgato.com/en/gaming/stream-deck
5. https://loupedeck.com/en/products/loupedeck-plus
6. https://upload.wikimedia.org/wikipedia/commons/b/b0/Serial_frame.png
7. https://www.circuito.io/

Literatura

[ADC()] Arduino: Analog measurements. URL http://meettechniek.info/embedded/arduino-analog.html.

[Ard()] Arduino leonardo datasheet. URL https://www.electronicsdatasheets.com/manufacturers/arduino/parts/arduino-leonardo.

[ard(a)] Getting started, a. URL https://www.arduino.cc/en/Guide/HomePage.

[ard(b)] Software, b. URL http://www.arduino.cc/en/main/software.

[vel(a)] Capacitive touch sensor switch, a. URL https://www.velleman.eu/products/view/?id=435524.

[vel(b)] Hc-sr05 ultrasonic distance sensor, b. URL https://www.velleman.eu/products/view/?id=435526.

[Malesevic(2018)] Ljubomir Malesevic. *Osnove elektrotehnike I*. Studij elektronike i elektroenergetike, 2018.

[the AREF pin(2020)] Arduino Tutorials the AREF pin. Tronix, 2020. URL https://tronixstuff.com/2013/12/12/arduino-tutorials-chapter-22-aref-pin/.

7 Popis priloga

Uz ovaj rad priložene su sljedeće datoteke:

- driver.py - driver skripta
- GUI.py - skripta grafičkog sučelja
- mainwindow.ui - datoteka za prikaz glavnog prozora u grafičkom sučelju
- config.json - konfiguracijska datoteka
- arduino_kod.ino - Arduino sketch