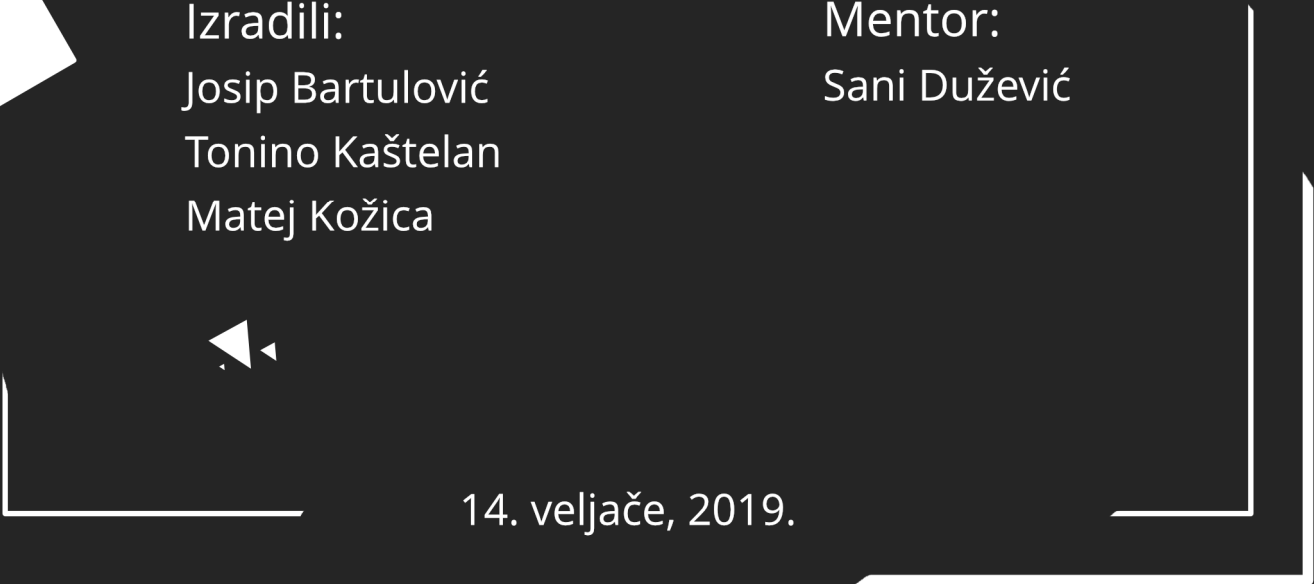




macro
TOUCH



Izradili:
Josip Bartulović
Tonino Kaštelan
Matej Kožica

Mentor:
Sani Dužević

14. veljače, 2019.

Sadržaj

O timu	4
Problem	7
ŠtoMacroTouchnudi?	8
O aplikaciji	11
Korištenje	13
Uređaj	13
Software	14
Sučelje	14
Tehnologije	32
Hardware	32
Software	38
Python	38
Java i Javascript	57
Zahvala	58



I.poglavlje

O TIMU

O timu

Naš je tim najlakše opisati kao skupinu zaljubljenika u IT. Zadivljeni svime što se događa danas u svijetu tehnologije, želimo postati dio toga i doprinijeti toj zajednici. Uvedeni smo u taj svijet putem naše škole, III. gimnazije Split, koja nam pruža potporu i resurse kako bismo razvijali svoje mogućnosti i želje. U takvom smo okruženju, želeći ostvariti doprinos, došli i na ovu ideju. Promatrali smo ostatak IT zajednice i primjetili problem s kojim se ona suočava; vrijeme izrade projekta. Kako bi se IT područja svela na što manje tipkanja i što više na same izvedbe projekta. Razni proizvođači su pokušali pronaći rješenje. No, uvidom da su neka rješenja preskupa ili neoptimalna, došli smo na ideju za MacroTouch.

Ekipa se sastoji od troje maturanata III. gimnazije Split. Svi dijelimo istu strast prema programiranju, uz razlike u vremenu razvitka te zanimacije. Vođa projekta je najduže u području IT-a. Sa svojim je predznanjem on pokrenuo interes ostatka ekipe za to područje i opskrbio ju potrebnim znanjem za početak.

Josip Bartulović

Voditelj ekipe koji se programiranjem bavi od sedmog razreda osnovne škole kada se po osnovnoškolskom programu upoznao sa programiranjem u BASIC-u pomoću kojeg je izrađivao jednostavne igrice. Nakon toga samostalno usvaja osnove Jave i u osmom razredu sudjeluje na svom prvom InfoKupu u kategoriji algoritama. U isto vrijeme pohađa

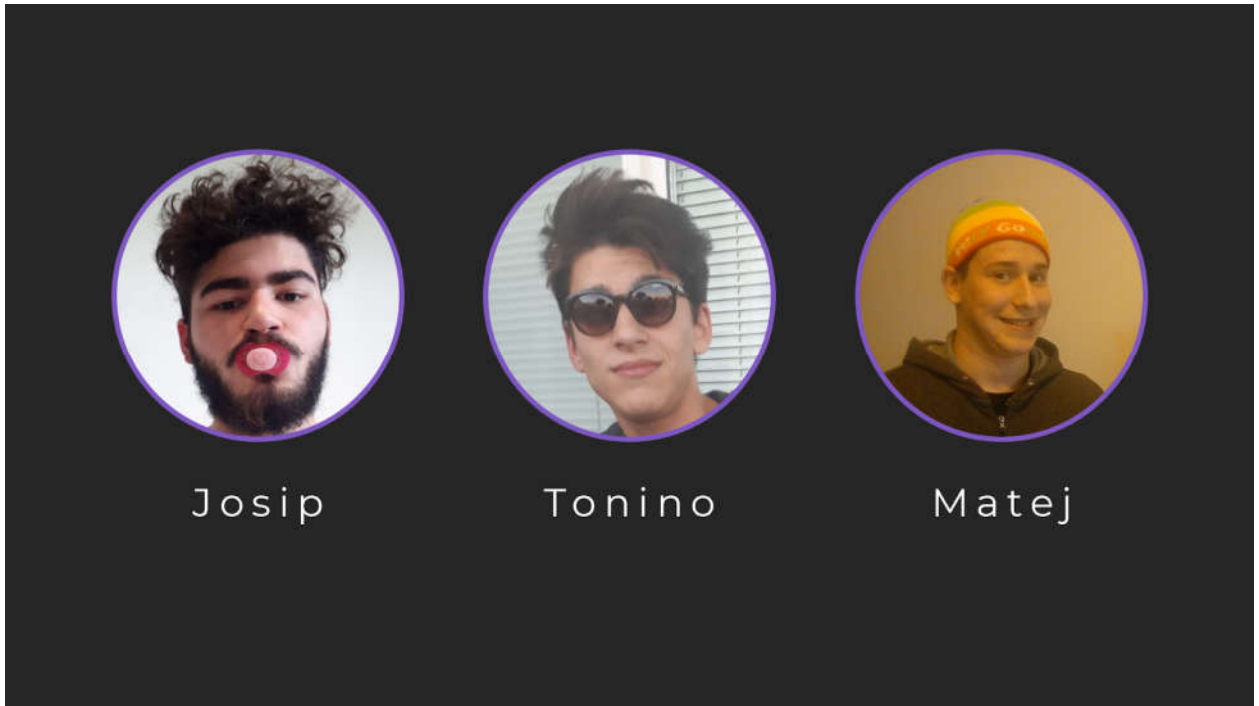
vannastavne aktivnosti *embedded* programiranja u Arduino. Nakon osnovne škole upisuje MIOC gdje se na poticaj profesorice iz informatike upoznaje s C-om, C++-om, a kasnije i Pythonom kojim se do danas služi. Tokom srednje škole redovito sudjeluje na natjecanjima RoboKup, InfoKup i Honi te u želji da nauči kako programiranje funkcionira “kod velikih” upisuje internship program udruge DUMP. Tu se susreće sa web developmentom i prihvaća programiranje kao disciplinu kojom se želi baviti cijeli život.

Matej Kožica

Član ekipe koji se IT-om počeo baviti u prvome srednje. Python ga je upoznao sa svijetom programiranja i nakon godinu dana vježbanja algoritama ušao je u svijet objektno-orijentiranog programiranja. Na DUMP Internshipu naučio je osnove C#, Entity frameworka te HTML i CSS. Nakon toga samostalno se uputio u svijet frontenda i naučio Java Script te React. Od malih nogu gajio je ljubav prema cretanju i dizajnu pa je svoje vještine ukomponirao sa znanjem na frontendu.

Tonino Kaštelan

Član ekipe koji je najkasnije ušao u svijet IT-a. Kako nije bio u istom razrednom odjelu kao ostatak tima, put u IT nalazi u Centru izvrsnosti gdje je i upoznao ostatak ekipe. Počeo je s web aplikacijama na frontendu s Reactom te je kasnije napravio tranziciju na backend. Na backendu se upoznao s Pythonom i frameworkom Flask. U Pythonu i C++ je naučio pisati algoritme. Ovaj projekt mu je postavio novi izazov u upoznavanju s *embedded* programiranjem unutar Pythona. Osim Pythona, poznaje i C#.



Slika 1. Ekipa



II.poglavlje

PROBLEM

Problem

Kako je navedeno, pred današnje tzv. *Content creatore* postavljeni su rokovi koje je više nego često, teško ispuniti. Nekad zato što je projekt kompleksan i zahtjevan, a nekad zato što je vremenski neizvediv, iako vrlo jednostavan za realizaciju. Većina aplikacija i programa ima prečace koji zamjenjuju korištenje miša, npr. Ctrl+B u Microsoft Office Wordu podebljava tekst. No kratica je puno, postaju sve složenije te svaka aplikacija ima svoj set kratica koje treba zapamtiti. Pamteći sve kratice i tipkajući ih ne gubi se puno vremena, no korisnicima bi bilo puno lakše kada ih ne bi trebali pamtiti i kada bi njihov unos bio puno brži.



Slika 2. Korištenje više tipkovnica zbog različitih makro konfiguracija



III.poglavlje

ŠTO
MACROTOUCH
NUDI?

Što MacroTouch nudi?

S ciljem da učinimo svakodnevno korištenje računala udobnijim, bržim i lakšim osmislili smo proizvod koji pruža nov i inovativan način interakcije sa računalom. Taj uređaj je MacroTouch. Njegov je smisao da vam na svom 7" zaslonu sve najbitnije funkcije pruži nadohvat ruke.

On se na vaše računalo spaja preko lokalne mreže, sinkronizira se sa aplikacijama koje koristite i njihovo grafičko sučelje vam pruža nadohvat ruke.



Slika 3.MacroTouch

Ima sposobnost zamjene osnovnih ulaznih jedinica poput miša i tipkovnice što ga čini savršenim za korištenje iz udobnosti kreveta i kauča.

One funkcionalnosti koje vama nedostaju na MacroTouchu možete sami napraviti. On vam daje mogućnost izrade gotovo beskonačnog broja makroa, neograničenih sekvenci tipkovničkih naredbi koje se mogu paralelno ili jedno za drugim izvoditi. S obzirom da gotovo sve aplikacije danas za funkcionalnosti grafičkog sučelja imaju ekvivalentnu kraticu na tipkovnici mogućnosti personaliziranja vašeg radnog prostora i vremena postaju beskrajne.

Postoji rješenje slično našem, zvano StreamDeck, no MacroTouch za razliku od StreamDecka nudi mnogo više mogućnosti. Slobode dodavanja velikog broja konfiguracija, korištenje raznih widgeta, toolbara od aplikacija itd.



Slika 4.StreamDeck uređaj

Ostala rješenja (poput Logitechovih tipkovnica) kao i StreamDeck nailaze zapravo na isti navedeni problem, ograničenost. Cilj u stvaranju

MacroToucha bio je stvoriti uređaj koji će korisniku dati slobodu upravljanja njime na koji god način želi. Stoga naš uređaj nije ograničen fizički, već samo željama i maštom njegovog korisnika. Dakle, naš uređaj za malu cijenu nudi ugodu u stvaranju novih sadržaja i realizaciji projekata kakvu ne nudi ništa na trenutnom tržištu.



IV.poglavlje

O APLIKACIJI



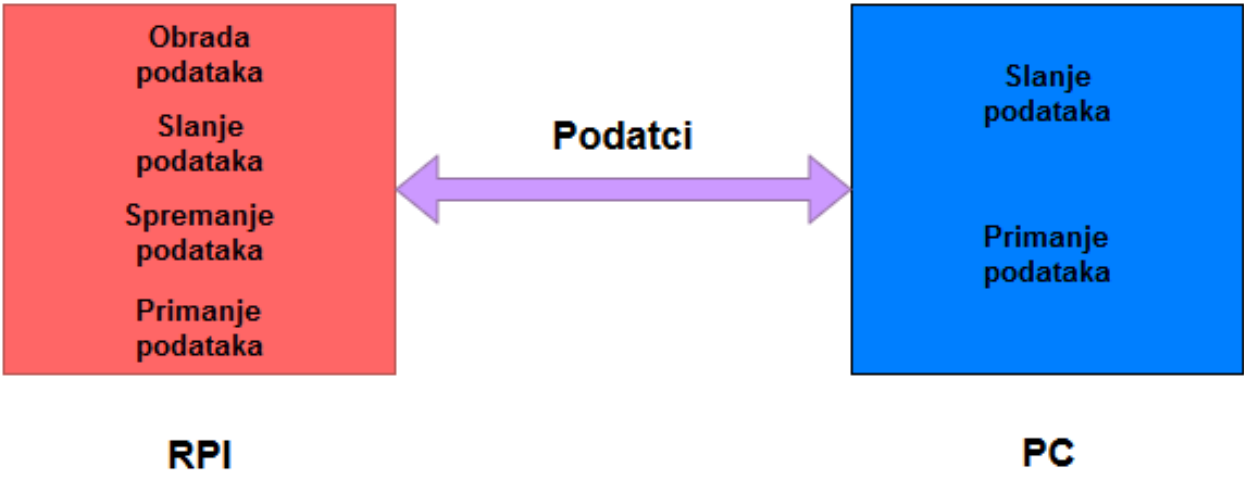
O aplikaciji

Pošto imamo fizički dva dijela, odnosno MacroTouch uređaj i korisničko računalo, tako imamo i dva dijela aplikacije za svaki fizički dio.

Računalo nema velik dio aplikacije, zapravo nema nikakvo grafičko sučelje već samo sluša podatke sa uređaja te ih obrađuje i vrši operacije ovisno zahtjevu. Pa tako na računalu imamo samo proces koji se vrti u pozadini.

S druge strane, na MacroTouchu, se dešava sva logika programa. MacroTouch obrađuje sve podatke koje je korisnik odabrao (primjerice makro naredba) u željen oblik, te ih šalje na desktop server. Tako se ostvaruje komunikacija između naša dva dijela aplikacije. Osim što Raspberry šalje podatke na desktop, on je ujedno onaj koji radi teži posao kod povezivanja. Pretražuje lokalnu mrežu, nađe ip adresu računala te se tada međusobno povežu.

Ostavljanjem težeg dijela posla na Raspberry Pi, ciljali smo na rasterećenje korisničkog računala i nesmetanog izvođenja naredbi na desktopu u što kraćem vremenu što smo tako i uspjeli.





V.poglavlje

KORIŠTENJE



Korištenje

Uređaj

MacroTouch nema kompleksno hardversko sučelje. Cilj je bio napraviti uređaj koji je intuitivan za korištenje u svakom smislu. Tako smo na njega kao fizičku tipku jedino postavili On/Off prekidač.



Slika 5. On/Off prekidač

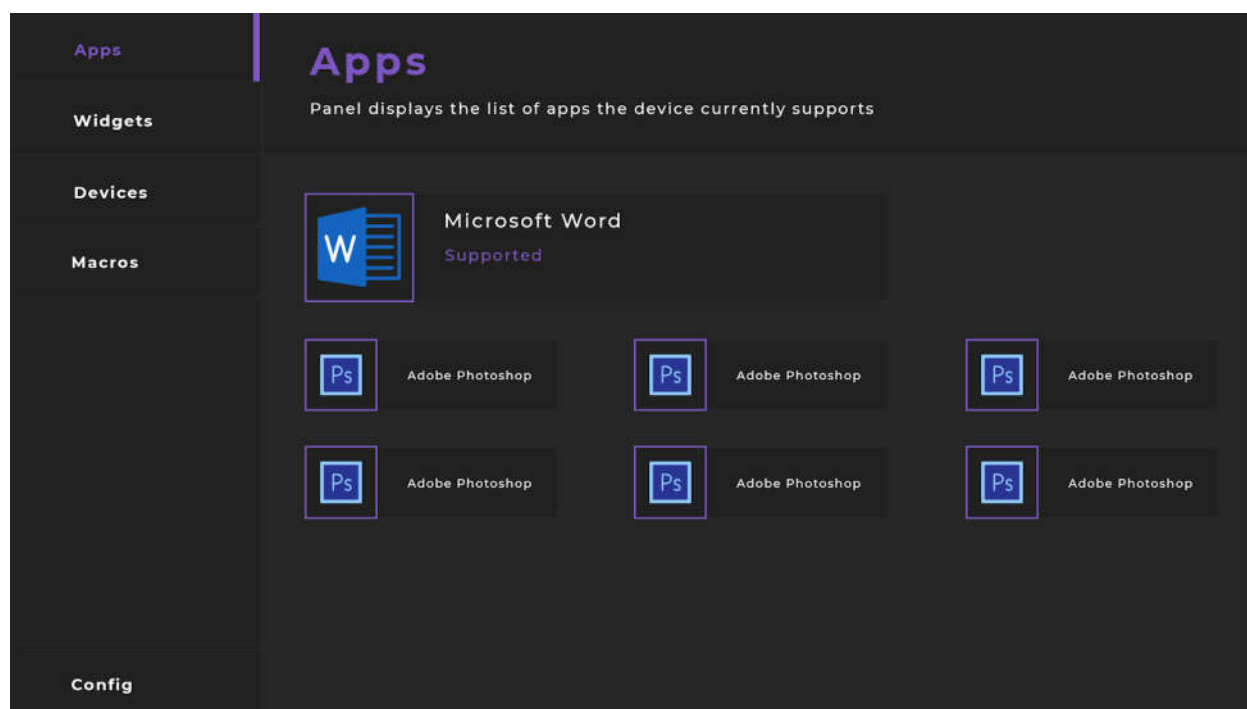
Pokreće ga baterija, stoga zahtijeva punjenje. Kako bi se mogao puniti, s vanjske strane ima utor za micro USB kabel. Plan je kasnije napraviti tranziciju na USB tip C, iz razloga što on polako zamjenjuje micro USB. Ta dva elementa su zapravo jedina koja trebaju za korištenje hardverskog dijela.

Software

SUČELJE

Apps

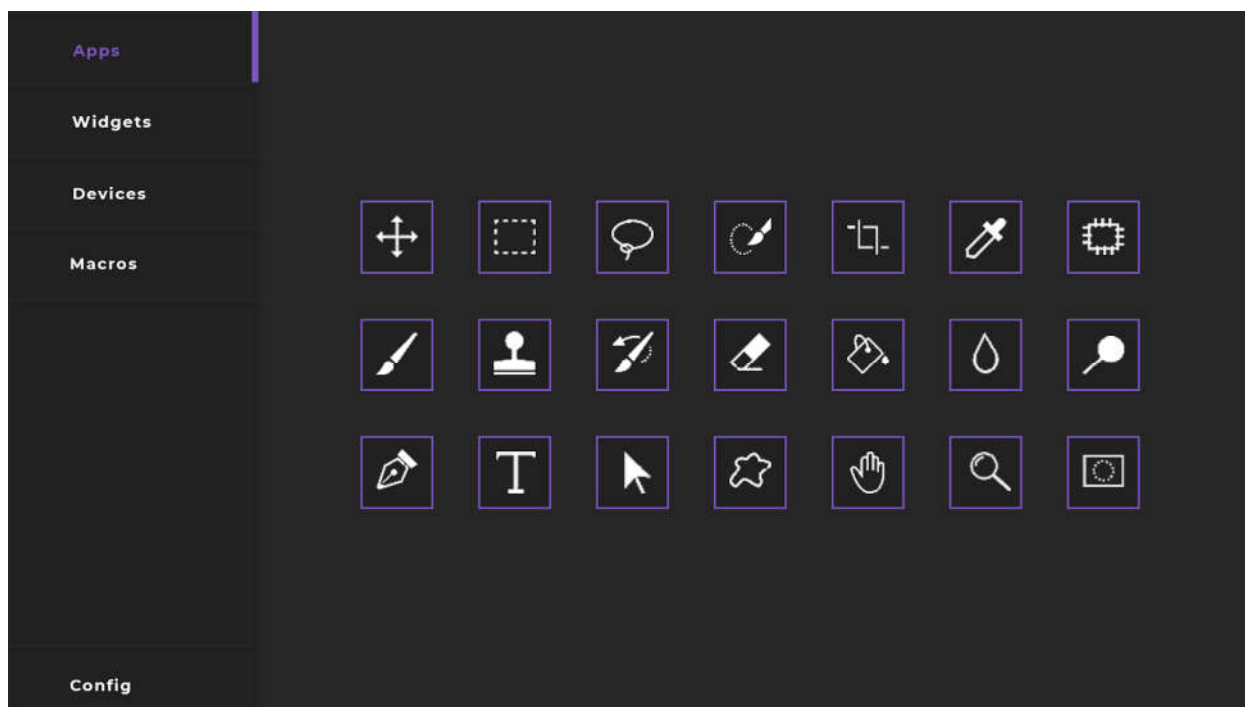
Apps kartica korisniku nudi aplikacije koje su podržane na MacroTouchu. Iznad same mreže ikonica, nalazi se aplikacija koja se trenutno koristi na računalu. Dodirom ikonice aplikacije koja se trenutno koristi na računalu pali se njeno sučelje na MacroTouchu. MacroTouch trenutno podržava Adobe Photoshop te Microsoft Office Word.



Slika 6. "Apps" kartica

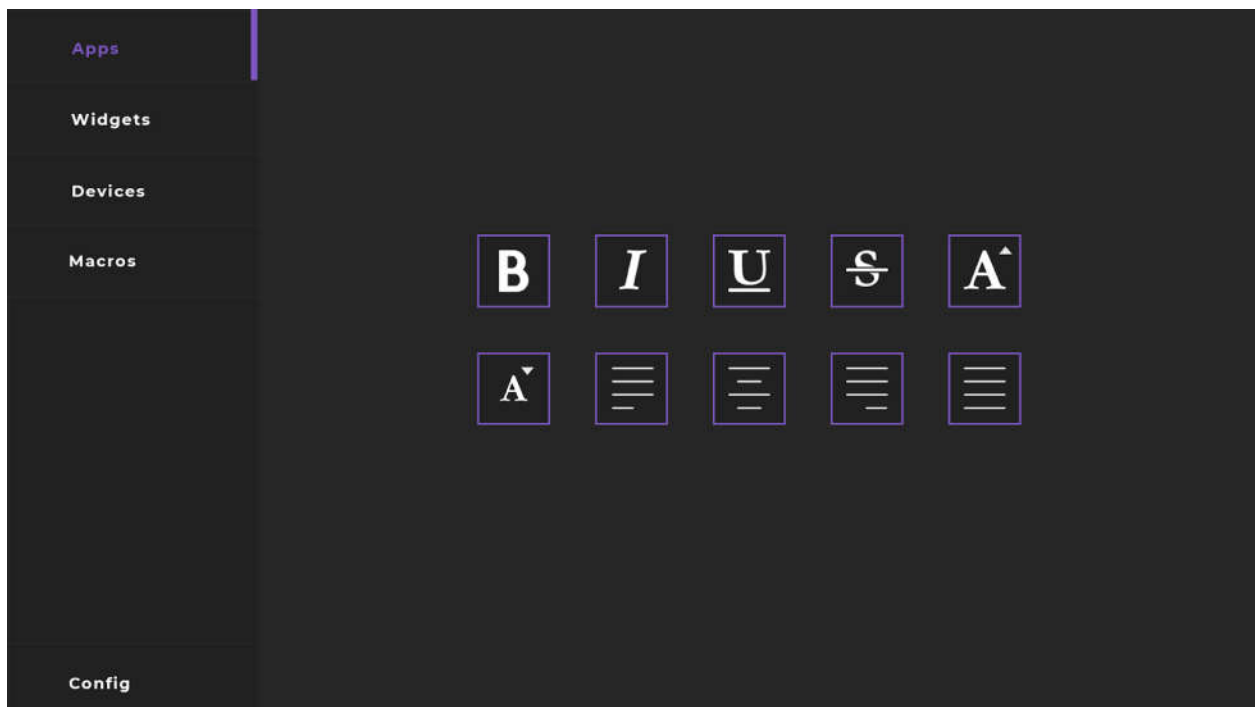
Adobe Photoshop sučelje nam u mrežnom rasporedu tipki predstavlja njegov klasični toolbar. Dodirom neke od ponuđenih tipki na MacroTouchu, na korisničkom računalu se aktivira pripadajuća naredba.

Mnogi dizajneri koristeći Adobe Photoshop moraju učiti njegove kratice i raditi vratolomije na tipkovnici kako bi neke od njih napravili. Pomoću Adobe Photoshoptoolbara na MacroToucha korisnici Photoshopa nadohvat ruke imaju sve najčešće korištene kratice i uz jedan dodir poziva se funkcija tih kratica.



Slika 7. Adobe Photoshop sučelje

Microsoft Office Word sučelje imitira Wordovu alatnu traku u kojoj su sve ikone poredane sistematično. Korištenje i rad ovoga sučelja analogno je korištenju Photoshopovog sučelja. Mnogim korisnicima Microsoft Office Worda dok pišu nezgodno im je koristiti i miš samo kako bi podebljali tekst ili ga centrirali. Uz pomoć Microsoft Office Word toolbara na MacroTouchu korisnicima su te kratice nadohvat ruke i uz jedan dodir ekrana njihov tekst je podebljan bez da su koristili miš što ubrzava pisanje.



Slika 8. Microsoft Office Word sučelje

Widgets

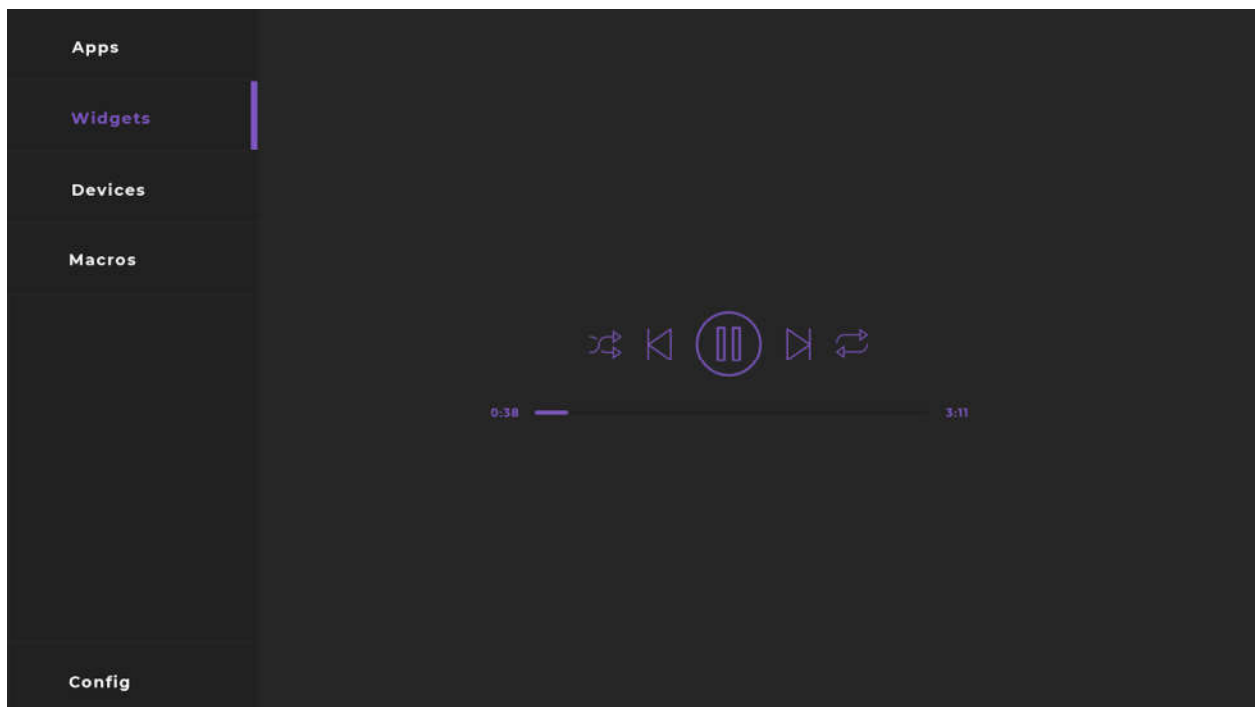
Widgeti su samostalne aplikacije koje ne ovise o računalu koje koristimo već su svojstvene za MacroTouch uređaj. Widgetima pristupamo klikom na karticu "Widgets" u glavnom izborniku. Widgeti su prikazani u obliku ikonica složenih u mrežu. Klikom na ikonicu palimo widget. Nakon što uđemo u widget klikom na 3 točkice na lijevoj strani ekrana vraćamo se na glavni izbornik. MacroTouch za sada nudi System Monitor, Music Player, Kalendar i Notes.

System Monitor je widget koji nam omogućava nadgledanje performansi našega računala tj. performanse procesora, grafičke kartice, radne memorije, diskova, interneta, ethernet i bluetootha.



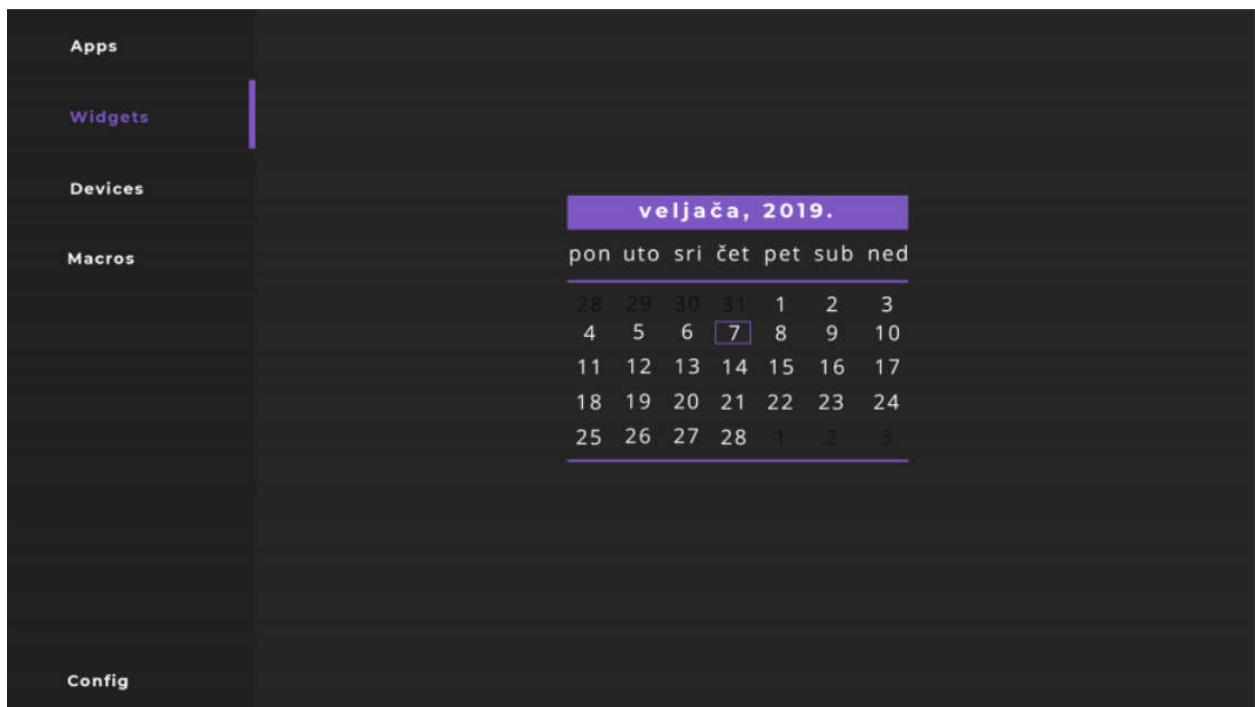
Slika 9. System Monitor widget

Music player omogućava pauziranje, prebacivanje na sljedeću tj. prošlu pjesmu, ponavljanje trenutačne pjesme te stavljanje playliste na shuffle. U budućnosti planiramo ovaj widget povezati sa playerima kao što su Spotify i Deezer tako da korisnik može vidjeti sliku albuma, ime pjesme, izvođača i albuma.



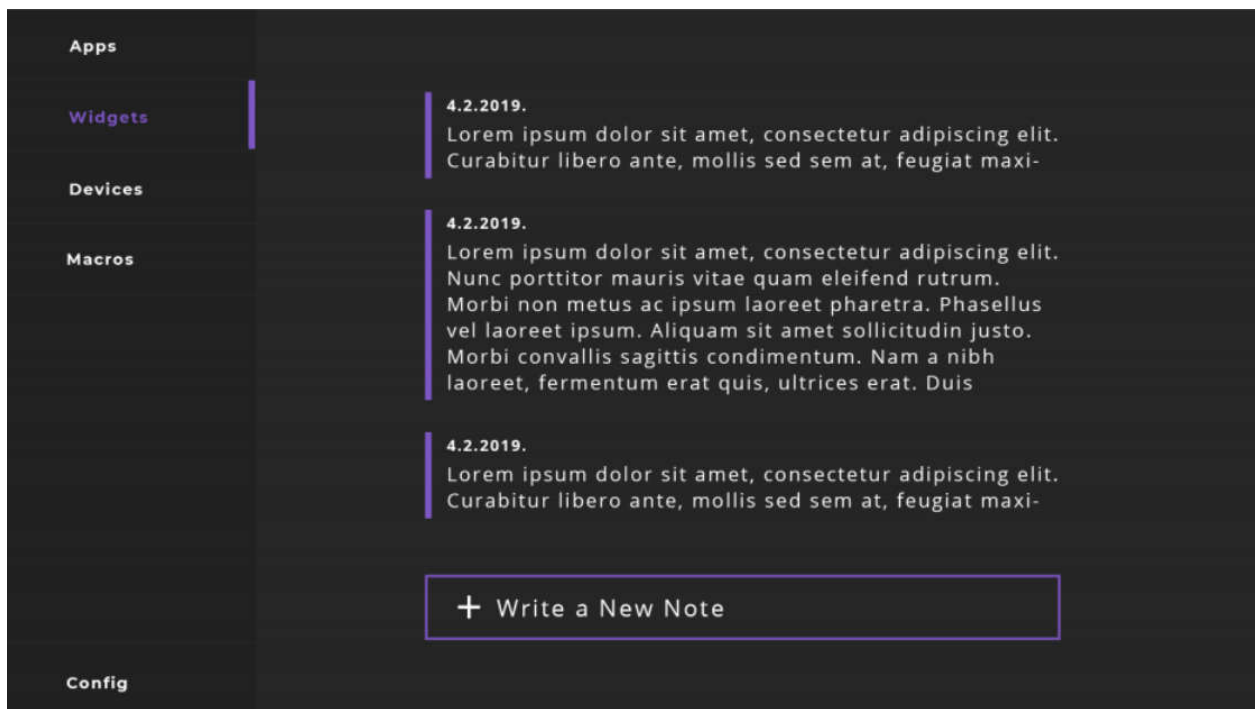
Slika 10. Music Player widget

Kalendar je widget na kojem možemo koristiti kalendar operacijskog sustava računala. U budućnosti kalendar planiramo povezati sa korisnikovim Google računom tako da može uređivati buduće događaje u kalendaru i podijeliti ih sa prijateljima.



Slika 11.Widget kalendara

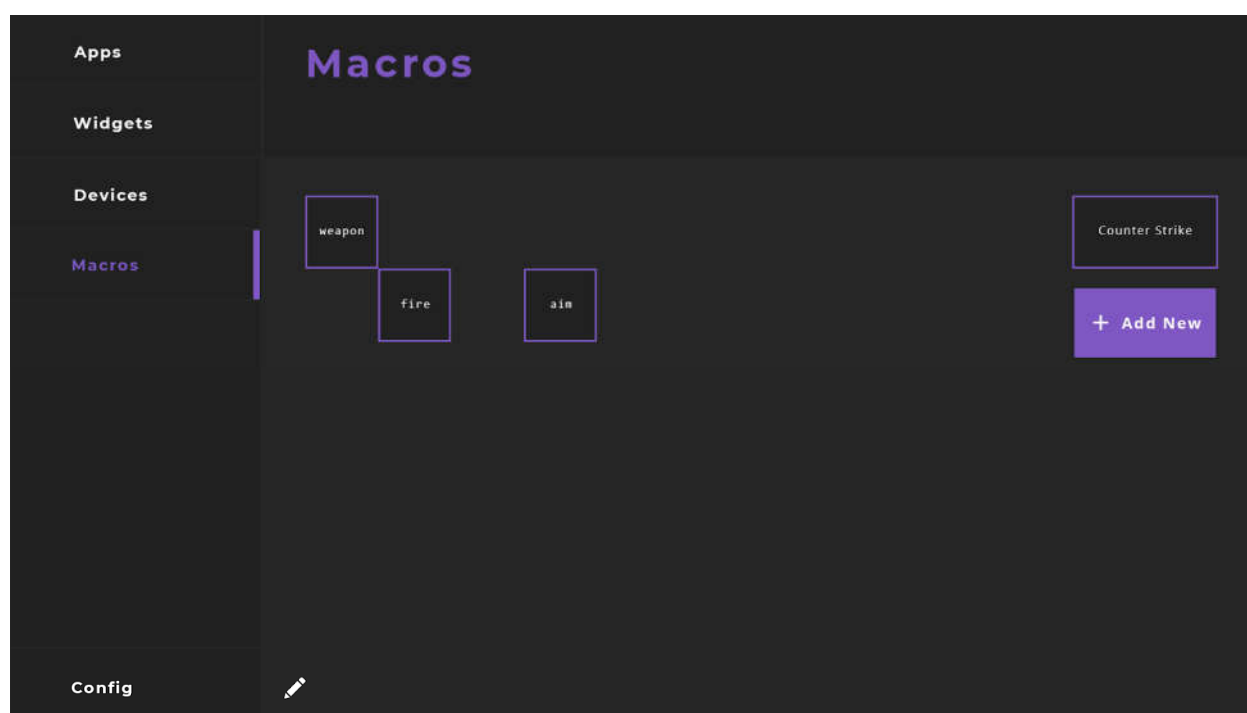
Notes je widget koji služi brzom zapisivanju zabilješki koje kasnije sprema u obliku JSON-a u bazu podataka. U budućnosti notes planiramo povezati sa korisnikovim Google računom kako bi mogao imati pristup Keep Notes te tamo spremati zabilješke.



Slika 12. Notes widget

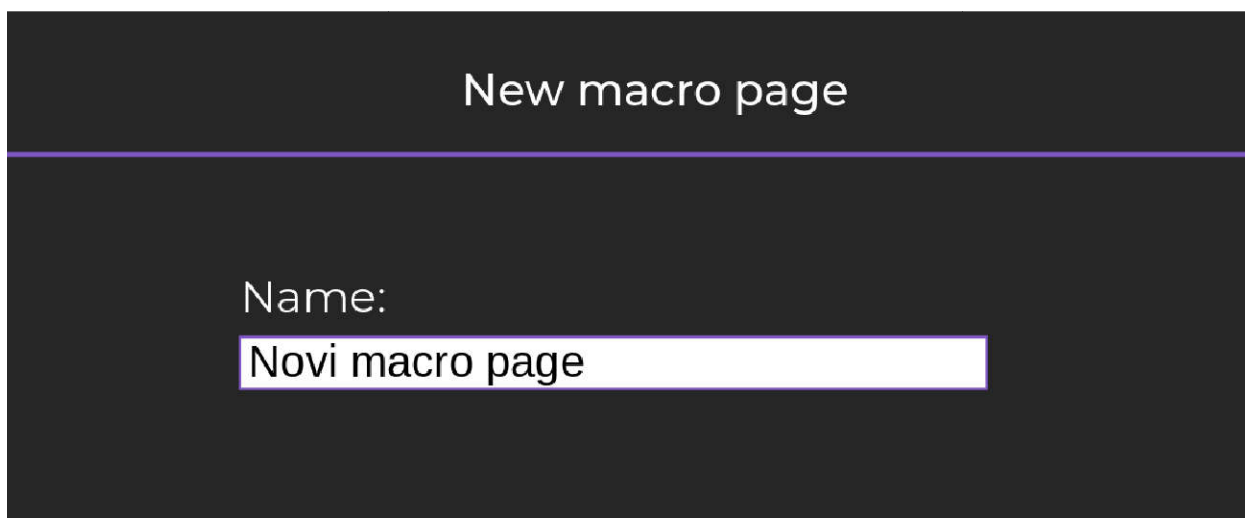
Macros

Kao što samo ime sugerira naša aplikacija nudi i korištenje makro naredbi. Makro naredba je slijed naredbi koje definiraju neku funkciju. Dodirom na karticu "Macros" korisnik pristupa makroima koji su prikazani u obliku ikona u mrežnom rasporedu. Pritiskom na ikonicu makro se aktivira te se zadana funkcija izvršava na računalu. Makro naredbe su u potpunosti dostupne na uređivanje kako bi ih korisnik mogao oblikovati prema svojim potrebama. Na primjer igrači World ofWarcrafta mogu napraviti makroe prilagođene igri tako da kombinaciju gumbova na tipkovnici mogu zamijeniti jednom tipkom u Macrotouchu.



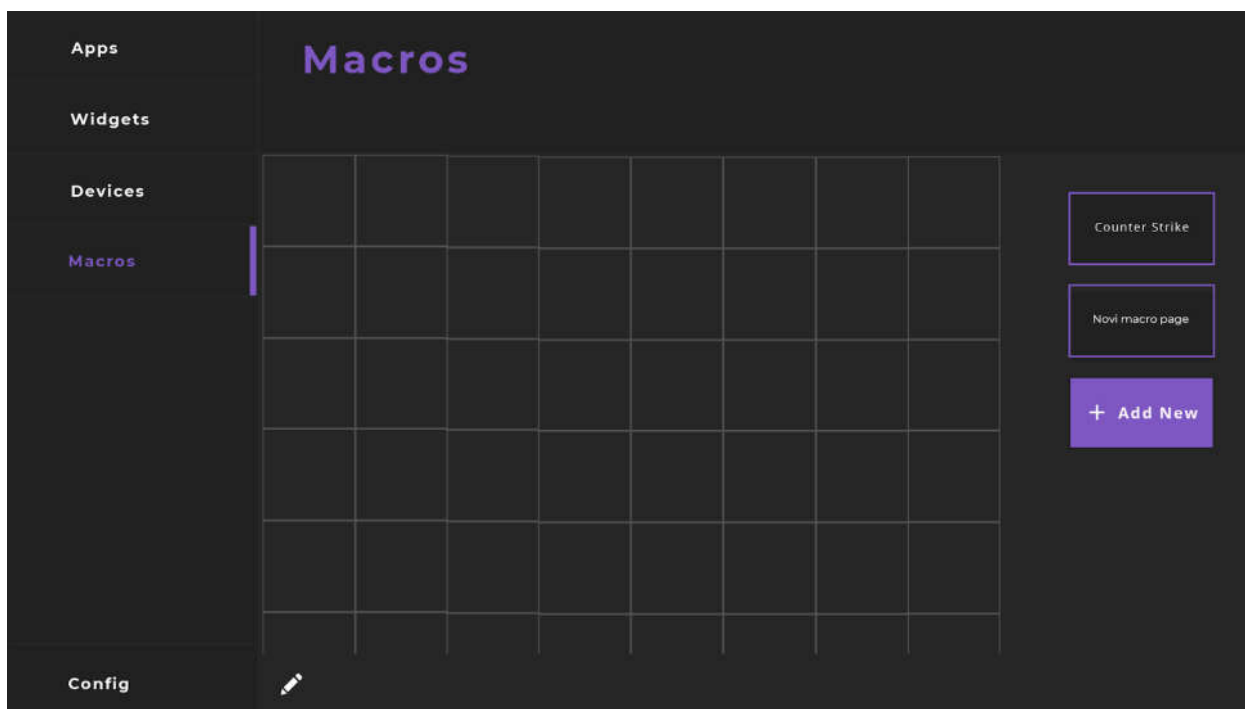
Slika 13. Kartica "Macros"

Kako bi dodali makro konfiguraciju potrebno je prvo dodirnuti [+ Add New] tipku. Ona otvara dijalog za unos imena. Unesite ime konfiguracije i potvrdite za nastavak.



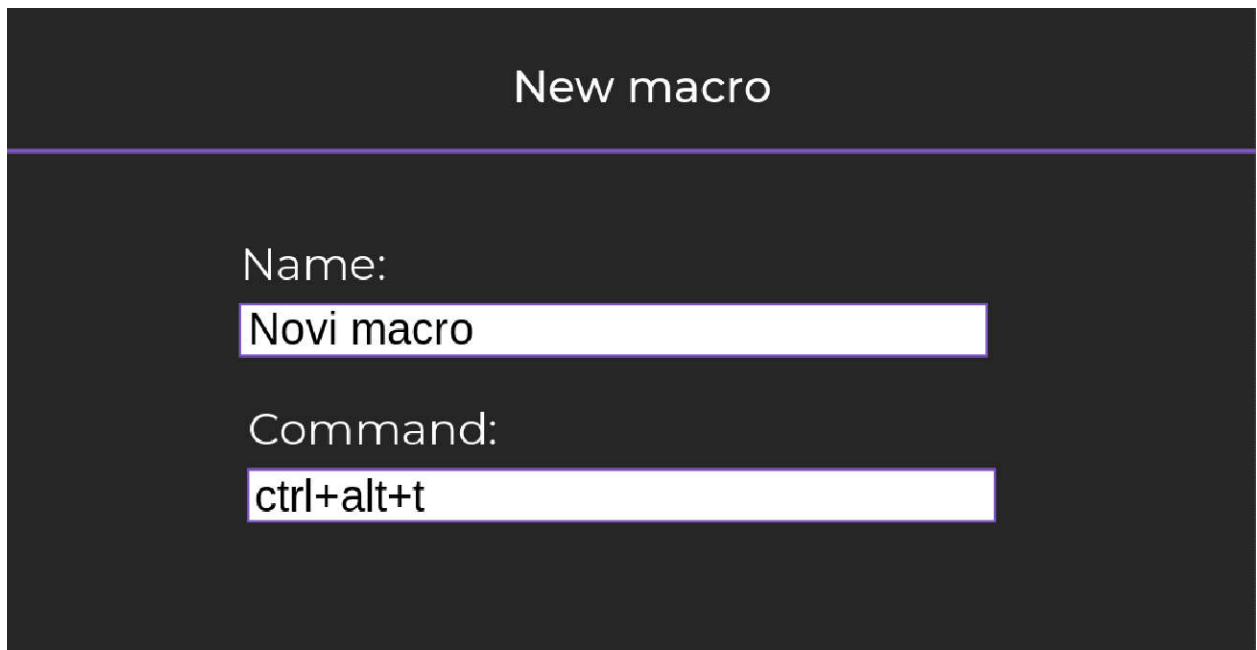
Slika 14. Unos imena konfiguracije

Nakon unosa imena konfiguracije, pred vama se otvara mrežasti raspored. Mjesta u mreži predstavljaju moguće pozicije makroa. Makro naredbe ne moraju nužno ići redom s lijeva na desna već mogu biti pozicionirana na bilo koje mjesto u mreži.



Slika 15. Grid

Kako bi pozicionirali makro, dodirnite neko prazno mjesto u mreži. Nakon dodira otvara vam se mogućnost unosa nove makro naredbe. Pri unosu: Pri odjeljivanju naredbi, neka separator bude '+' (bez apostrofa). Možete dodati i više makro naredbi po konfiguraciji.



New macro

Name:
Novi macro

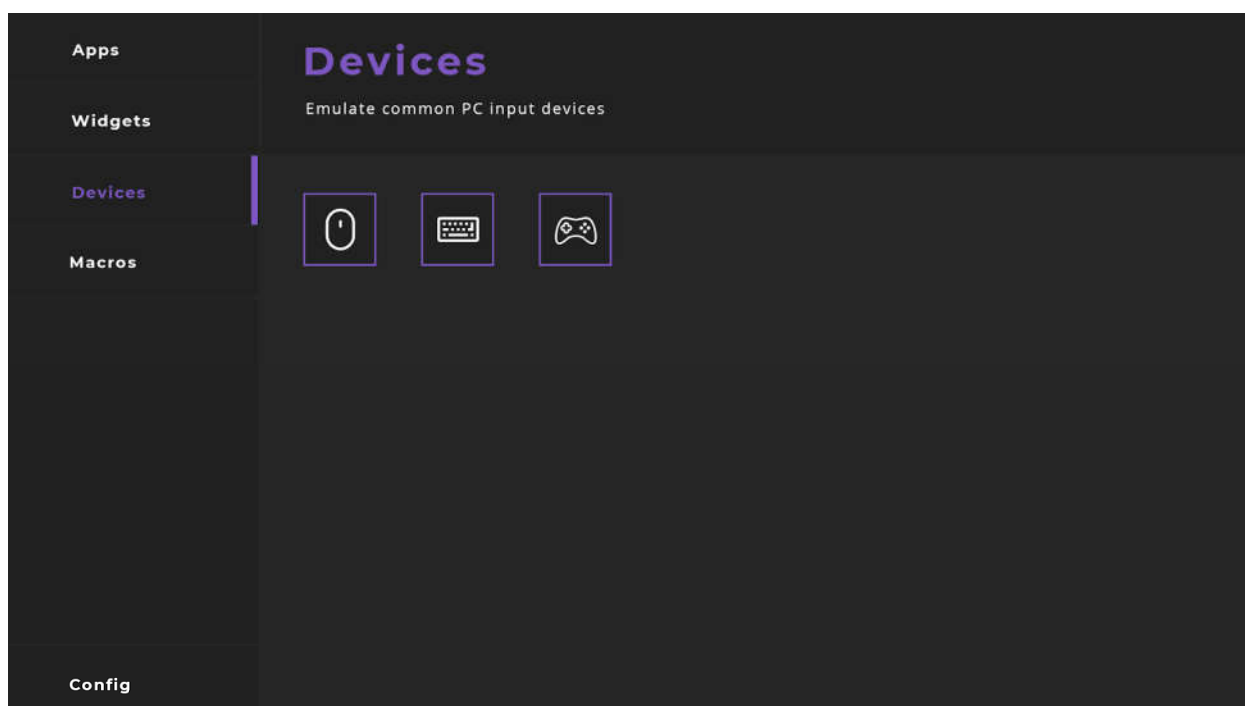
Command:
ctrl+alt+t

Slika 16. Unesi makro

Devices

Htjeli smo što više toga uklopiti u naš uređaj, stoga smo dodali i opciju korištenja MacroToucha kao neki od osnovnih ulaznih uređaja. Što znači da se može koristiti umjesto miša, (točnije trackpada) tipkovnice i joysticka. Time želimo korisniku omogućiti korištenje MacroTouch-a kao nešto više od samo radne platforme. To ga čini pogodnim za uporabu u situacijama kada računalo ne koristimo za radnim stolom, već iz udobnosti naših kreveta ili kauča.

Kako bi korisnik pristupio virtualnim ulaznim uređajima, mora pristupiti kartici "Devices" s lijeve strane ekrana uređaja. Nakon pritiska na devices, suočen je s izborom ulaznih uređaja poredanih u mrežnom rasporedu.

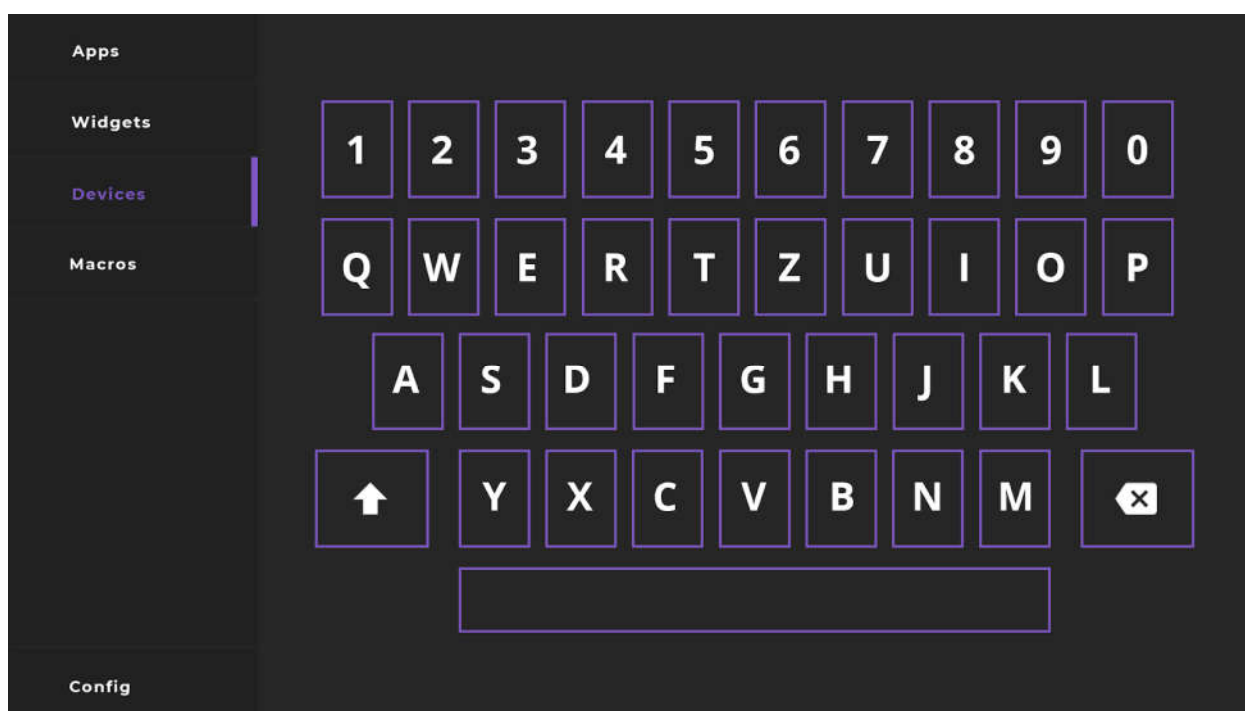


Slika 14. Kartica "Devices"

Kako bi korisnik pristupio virtualnom uređaju dovoljan je dodir na ikonicu željenog uređaja. Ispred korisnika tad se otvara sučelje tog uređaja.

a. Tipkovnica

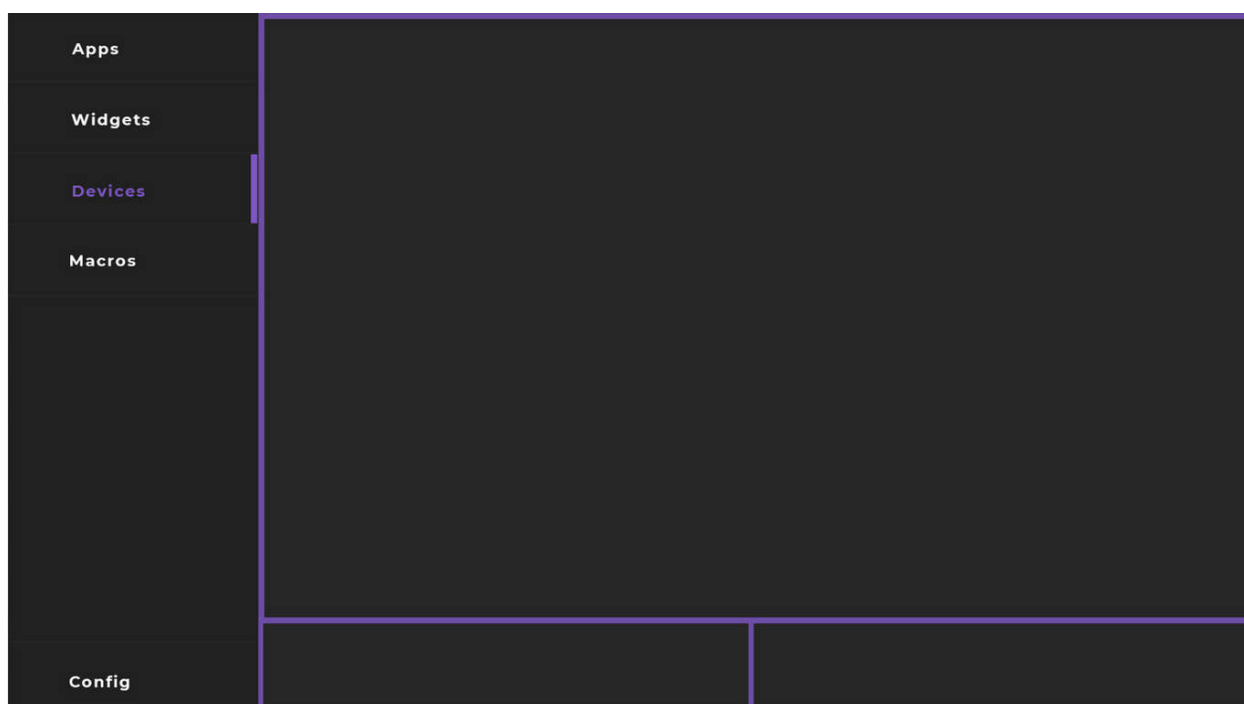
Sučelje tipkovnice isto je kao i ono klasične Android tipkovnice s kojom su svi korisnici već upoznati. Dodirom na željeni znak na korisnikovu računalu se u istom trenutku simulira pritisak tipke. Manjkavost ove tipkovnice je ta što ne omogućava pritisak na više tipki u isto vrijeme, ali to je nadoknađeno u tabu sa makroima.



Slika 15. Sučelje virtualne tipkovnice

b. Miš

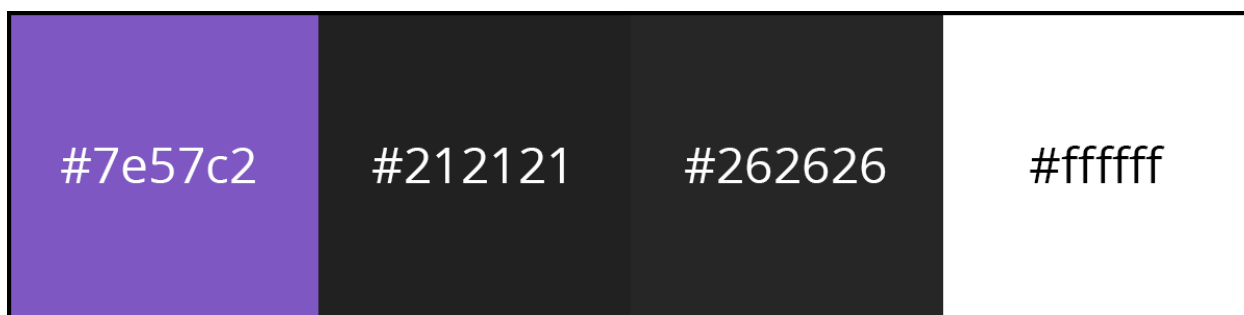
Sučelje za virtualni miš je siva površina sa dvije tipke. Kako bi pomakli miš na svom računalu, dovoljno je kliznuti prstom po sivoj plohi. Vrijedi spomenuti da ovaj miš ne radi poput običnog trackpada, prateći brzinu pokreta prsta, već računalu šalje poziciju vašeg prsta u odnosu na veličinu miša. To znači da ako želite pomaknuti miš ne trebate vući prstom po dodirnoj podlozi nego samo dotaknuti na poziciju na kojoj želite da se cursor računala pomakne. Za lijevi klik je dovoljno pritisnuti lijevu virtualnu tipku na sučelju za virtualni miš. Analogno vrijedi i za desni klik.



Slika 16. Sučelje virtualnog miša

Dizajn

Zbog popularnosti tamnijih tonova u modernim aplikacijama te ujedno iz razloga što tamnije boje na ekranima uzrokuju manje umaranje očiju, kao primarne boje aplikacije uzeli smo različite nijanse sive te svijetlo ljubičastu.



Slika 17. Paleta boja

Također smo i dizajn samog kućišta napravili u sličnim bojama. Odlučili smo se za minimalističan izgled grafičkog sučelja. Ujedno smo odabrali malu paletu boja zbog jednostavnosti i apela prema mlađim klijentima. Aplikacija koristi klasične elemente sučelja kako bi krivulja učenja korištenja aplikacije bila što manja. Glavne izborne kategorije poredane su u glavnom meniju sa lijeve strane. Ikonice aplikacija, widgeta, uređaja i makroa poredane su u mrežni raspored kako bi im korisnik lakše pristupio. Tok aplikacije napravljen je s time na umu da korisnik u 2 dodira može doći do bilo kojeg dijela aplikacije. Logotip je dizajniran na način da bude jednostavan i prepoznatljiv te da ukratko opisuje samu aplikaciju. Ikonice korištene u aplikaciji izradili su korisnici web stranice Flaticon.



VI.poglavlje

TEHNOLOGIJE

Tehnologije

HARDWARE

Raspberry Pi

U pozadini uređaja nalazi se mikro računalo Raspberry Pi. Iako nešto veće od plika karata dovoljno je snažno za potrebe MacroTouch-a. Pokreće ga 64-bitni četverojezgreniprocessor brzine 1,4 GHz, armchip koji je danas često dostupan na tržištu i nalazi se u mnogim pametnim telefonima. Ima 1 GB SDRAM-a, a kao trajna memorija koristi se memorijska kartica. Prije samog korištenja potrebno je na nju *zapržiti* jedan od operacijskih sustava koje RaspberryPi procesor može pokrenuti. Najpopularniji među njima su Raspbian i Windows 10 IoT.

Raspbian je GNU/LINUX operacijski sustav. Točnije derivat Debiana. Mogućnosti koje dobivamo instalacijom ovakvog operacijskog sustava su već instalirana podrška za većinu programskih jezika, jednostavna nadogradnja i konfiguriranje cijelog sustava.

Windows 10 IoT jednostavnija je i manja verzija windowsa. Namijenjen je korisnicima koji nisu upoznati sa radom na GNU/LINUX operacijskim sustavima koji zahtijevaju poznavanje rada u konzoli. Njima pruža već poznato sučelje u kojem mogu lako doći do stvari koje su im potrebne. Iako je manji i jednostavan za razliku od običnog Windows 10, svejedno je

više kompleksan od Raspbiana i preporuka za veće i ozbiljnije projekte na Raspberryu ga nije preporučeno koristiti.

Moguće je spojiti ga na internet preko wifi-a ili ethernet kabela. Ima podršku za bluetooth protokol te sadrži 4 USB ulaza za spajanje ulaznih i izlaznih jedinica. Također ima HDMI i audio izlaz što omogućava spajanje na vanjski monitor i zvučnik. Osim tih klasičnih metoda komunikacije ima i 40 programabilnih GPIO (general purpose input output) pinova na koje možemo spojiti senzore za: pokrete, svjetlinu, dodir, protok vode, otpor struje... Vrijednosti tih senzora možemo dohvatiti u programu na našem raspberry-u te ih koristiti u izradi raznih aplikacija.



Slika 18. Svjetlosni senzor

Osim senzora možemo spajati i razna trošila poput: LED dioda, motora, aktuatora i displaya. Protok struje kroz njih možemo također programski kontrolirati. Sve ove specifikacije čine Raspberry Pi izvrsnom platformom za izradu IoT (internet of things) uređaja, a to nam dokazuje i činjenica da se nalaze i u splitskim pametnim klupama.



Slika 19. Pametna klupa

Touchscreen

Uređaj se koristi pomoću 7 inchnogtouch displaya koji je na Raspberry spojen preko HDMI kabela preko kojeg se prenosi slika i USB kabela koji prenosi signale dodira i napaja električnom energijom display.



Slika broj 20. 7" ekran koji koristimo

Građa

Građa je možda bila najveći problem. Trebalo je staviti Raspberry Pi, napajanje i sve kabele koji idu uz to u jednu malu kutiju. Optimalno rješenje s te strane još nije postignuto, jer smatramo da možemo još smanjiti visinu MacroToucha. To namjeravamo putem promjene na Raspberry Pi Zero-a i drukčijeg HDMI kabela koji neće zauzimati toliko mjesta. Planiramo s trenutnih 225 x 140 x 40mm doći na upola niže kućište.

Trenutno, naš je prvi prototip građen od 3D printa plastike te drva. Namjera nam je imati aluminijsko kućište i drvenu konstrukciju sa strane. To je bio cilj od početka, no sad nismo bili u mogućnosti realizirati takvo nešto najviše iz financijskih razloga. No osim materijala, izgled će ostati približno isti. Što se tiče funkcionalnosti, omogućiti će pasivno hlađenje uz rupice s donje strane uređaja. Kako ni drvo ni aluminij nisu teški uređaj će biti prenosiv, no ujedno i izdržljiv. Sa strane dizajna ciljali smo na moderan i prirodan dizajn koristeći kombinaciju metala i drva.



Slika 21. Materijali koje namjeravamo koristiti

S unutarnje strane, podijelili smo kućište, na dva dijela; dio s baterijom i kabelima, te dio s Raspberryem. Sve unutra je dobro pričvršćeno kako nebi došlo do loma prilikom prenošenja. Stoga prototip sam trenutno nije robustan i otporan no finalni produkt će biti čvrst i moći će se koristiti bez ikakvog straha od loma.



VII.poglavlje

SOFTWARE



Software

PYTHON

Kivy

Kivy je opensourcePython biblioteka za brz razvoj aplikacije kojima je potrebno moderno i intuitivno grafičko sučelje. Posebno je usmjereno na aplikacije koje kao glavni način interakcije koriste dodir poput tablet i mobilnih aplikacija, ili u našem slučaju IoTsustava.

Osim što je opensourcekivy je napravljen s ciljem da ga mogu pokrenuti svi veliki operacijski sustavi: OS X, Android, Windows, GNU/LINUX, iOS. Uz to rad na većem broju platformi ne zahtijeva nikakvo mijenjanje koda.

Besplatan je za korištenje te je pod MIT-ovom licencom. Razvija ga tim profesionalaca i koristi se u izradi komercijalnih proizvoda. Framework je stabilan, ima dobro dokumentiran API i vodič za programiranje koji uvelike olakšava proces učenja. Jednostavan za korištenje programerima svih razina znanja.

```
fromkivy.appimport App
fromkivy.uix.buttonimportButton

classTestApp(App):
    defbuild(self):
        returnButton(text='Hello World')

TestApp().run()
```

Izrađen je preko OpenGL ES₂ grafičke biblioteke. Dolazi sa više od 20 već integriranih *widgeta*. Pisan je u C programskom jeziku pomoću Cython-a koji kompilira Python u C.

Izrada aplikacija u Kivyu uvelike je olakšana korištenjem njegovog jezika posvećenog izradi grafičkog sučelja i interakcije između pojedinih komponenti ili kako ih Kivyframework naziva - widgeta. Svaki widget je samo instanca Python klase koja nasljeđuje od nekog od već napravljenih widgeta.

Primjer definiranja i korištenja widgeta u Kivy jeziku

```
<BotunKlasa@Button>: #Izrada nove klase botuna pomoću kivy jezika
    text: "Text botuna"

BotunKlasa: #Instanciranje napravljene klase
```

Primjer definiranja i korištenja widgeta u Pythonu

```
fromkivy.appimport App
fromkivy.uix.buttonimportButton

classBotunKlasa(Button):
    text = "Text botuna"

classNovaApp(App):
    defbuild(self):
        returnBotunKlasa()
```

Jezik je vrlo sličan CSS-u. Sastoji se nekoliko glavnih komponenti:

- **Pravilo**

Slično CSS pravilu, Kivy pravilo se pridjeljuje određenom widgetu unutar stabla widgeta i modificira njegova svojstva. Pravila se koriste za dodavanjem grafičkih svojstava (npr. podloga za crtanje ili orijentacija

elemenata na ekranu) ili dodavanjem pravila za interakciju prilikom određenog eventa (npr. funkcija koja se pokreće klikom na botun)

- **Prvi widget u stablu (tzv. rootwidget)**

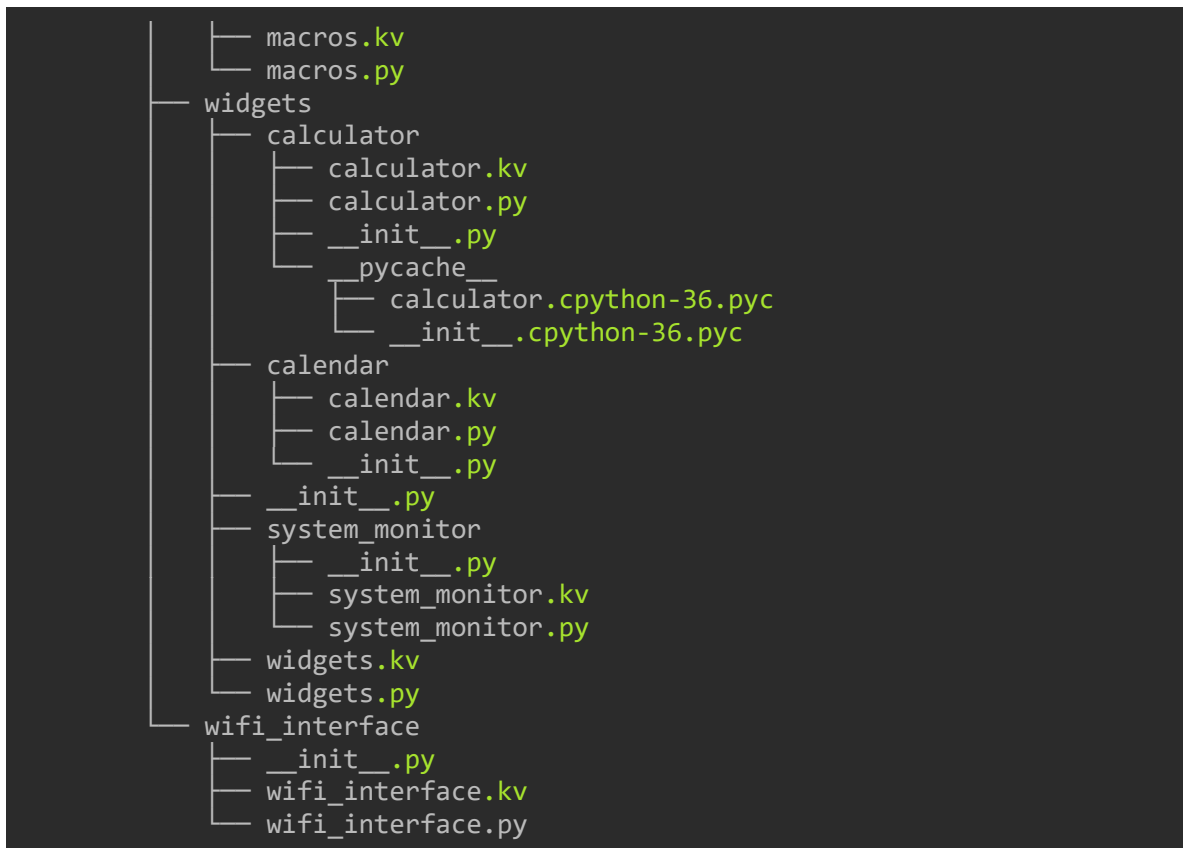
Aplikacija mora sadržavati barem jedan rootwidget zbog toga što, slično kao i web stranica, ima strukturu stabla.

- **Dinamičke klase**

Omogućavaju izradu novih widgeta bez njihove deklaracije unutar Python koda.

Struktura koda:





Raspberry je glavni folder kivy dijela aplikacije u kojem su strukturirani svi ostali Python paketi. Svaki paket sadrži glavne .py i .kvfileove. U .kv fileu definirani su svi widgeti te njihov izgled, a u .pyfileovime funkcionalnosti i interakcije koje međusobno vrše.

App.py definira glavnu klasu, tj prozor aplikacije, njene dimenzije. Definira funkciju za izmjenu prikazanih elemenata te svim panelima prosljeđuje funkcije potrebne za komunikaciju sa računalom.

Unutar foldera common nalaze se widgeti koji se koriste kroz veći broj .kvfileove. To je napravljeno sa željom pisanja što manje kolicineredudantnog koda.

Panels direktorij sadrži sve glavne panele aplikacije koje se izmjenjuju klikom na jednu od trakica na izborniku. Svaki od panela sadrži pod pakete u kojima su svi elementi koji se nalaze unutar panela. Tako devices ima pakete za miš, tipkovnicu i joystick, widgets ima system monitor notes i kalendar, apps ima paket za svaku od podrzanih aplikacija

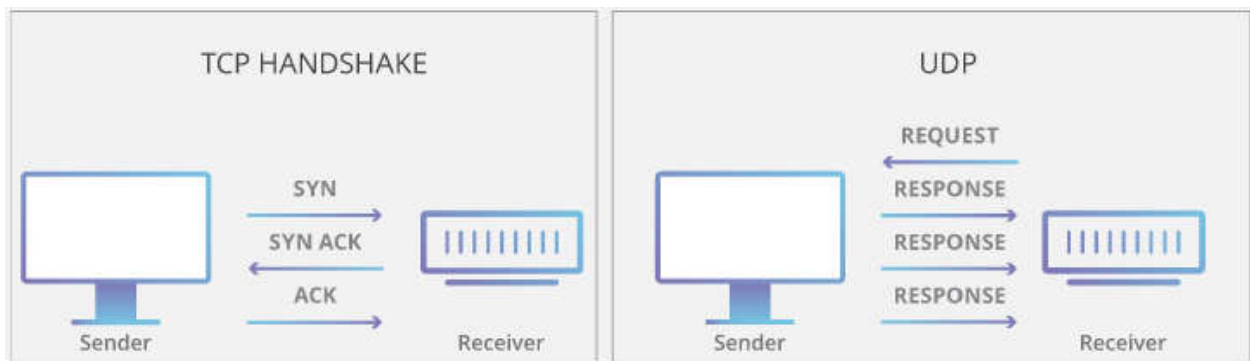
Fonts sadržava downloadane korištene fontove, a icons sve potrebne slike i ikone. To se nalazi unutar samo aplikacije kako bi smanjili vrijeme učitavanja prilikom pokretanja uređaja.

Unutar data.json file se spremaju svi potrebni trajni podatci. Za sada su to svi postojeći makroi i notesi koje korisnik zapiše. Nismo htjeli korsititi bazu podataka kako bi smanjili broj procesa koji se na uređaju stalno pokreće. Time produljujemo radno vrijeme samog uređaja i radni vijek baterije.

Sockets

Imali smo dvije opcije na umu za komunikaciju MacroToucha s korisničkim računalom, Bluetooth i putem lokalnog Wi-Fi-a. Trenutačno uređaj radi opcijom lokalnog Wi-Fi-a, no u budućnosti planiramo razviti i Bluetooth komunikaciju u slučajevima kada korisniku internetska mreža nije dostupna. Da bi imali komunikaciju putem lokalne mreže, potrebni su nam socketi.

Socketi su zapravo način komunikacije dva elementa koji se nalaze na istoj mreži. Rade tako da jedan socket sluša i čeka podatke dok mu ih drugi šalje. Drugim riječima, jedan socket se ponaša kao server dok je drugi klijent koji mu šalje podatke. Postoje dva tipa socketeta, TCP (TransmissionControlProtocol) i UDP (UserDatagramProtocol). TCP sockete nalazimo u web komunikaciji te su oni zapravo oslonac internet komunikacije. UDP socketi se koriste rjeđe u web komunikaciji, odnosno koriste se većinom u slučajevima kad je potreban tzv. *Real time messagingflowprotocol*. U našem slučaju, koristili smo UDP sockete. Takvi socketi, za razliku od TCP socketeta, ne trebaju potvrdu veze, te iz tog razloga nije sigurno hoće li svi podaci doći na server no to nadoknađuju svojom brzinom. To nam je mnogo važnije, jer povremeno izgubljeni podatak nije toliko važan koliko i brzina slanja tih podataka.



Slika 22. TCP socket vs. UDP socket

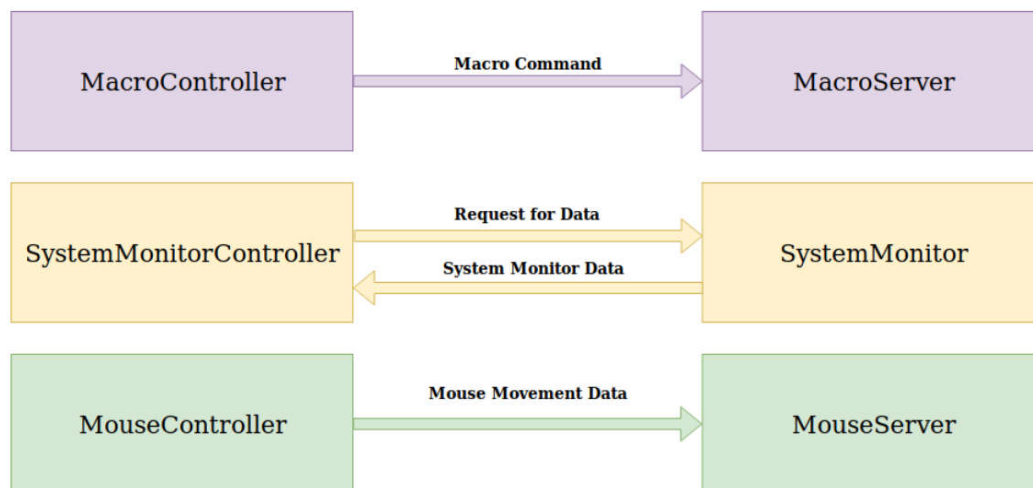
Za socket komunikaciju u našem projektu, koristili smo Pythonov istoimeni paket *socket*. Putem tog paketa, Python na vrlo jednostavan način omogućava komunikaciju putem socketa. Paket smo koristili na sljedeći način:

- **import socket**- omogućava korištenje paketa socket u našem kodu
- **socket.socket(socket.AF_INET, socket.SOCK_DGRAM)** - stvara objekt socketa koji koji spada u obitelj AF_INET i tipa SOCK_DGRAM
- **socket.sendto(data, address)** - socket šalje objekt data, odnosno podatke, na dani argument adrese
- **socket.bind(address)** - socket se veže na danu adresu i prima podatke koji dolaze sa te adrese
- **socket.recvfrom(size)** - prima podatke do dane veličine sa adrese na koju je socket vezan

AF_INET definira obitelj adresa gdje je hoststring koji predstavlja ime neke adrese ili ip adresu, a port je integer koji definira port na zadanoj ip adresi. SOCK_DGRAM postavlja tip socketa na UDP socket. Argument *data* u

metodi `sendto()` je tipa `bytes`, dok je argument `address` tipa `tuple`. Argument `size` je tipa `integer` i on definira velicinu podataka u bytesima.

Naša struktura komunikacije izgleda ovako:



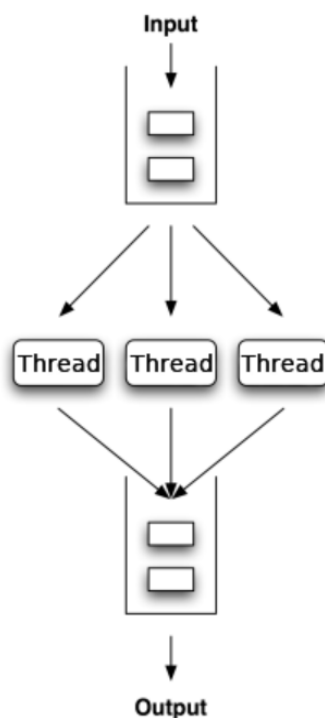
Slika 23. Struktura komunikacije u projektu

Pošto komunikacija između socketa nema različite vrste zahtjeva, a nama su bili potrebni, riješili smo to na svoj način. Podaci koji se šalju su tipa *dictionary*. Svaki podatak koji se šalje u sebi ima ključ *type* postavljen na tip zahtjeva u kojem se šalje taj podatak. Kad socket koji sluša primi podatak, obradi ga ovisno o tipu requesta kojeg pročita iz navedenog ključa.

Threading

Kao što smo napomenuli, koristimo sockete za komunikaciju. Pa tako imamo u programu sockete koji jako često primaju podatke. Tu smo se suočili s problemom. Naš uređaj treba imati vrlo brzu komunikaciju s korisničkim računalom, a uz jako česte zahtjeve i tada obrade podataka, dolazi do usporavanja. Također postoji mogućnost da se izgubi podatak za vrijeme obrađivanja drugog podatka. Stoga smo shvatili da su nam potrebni threadovi.

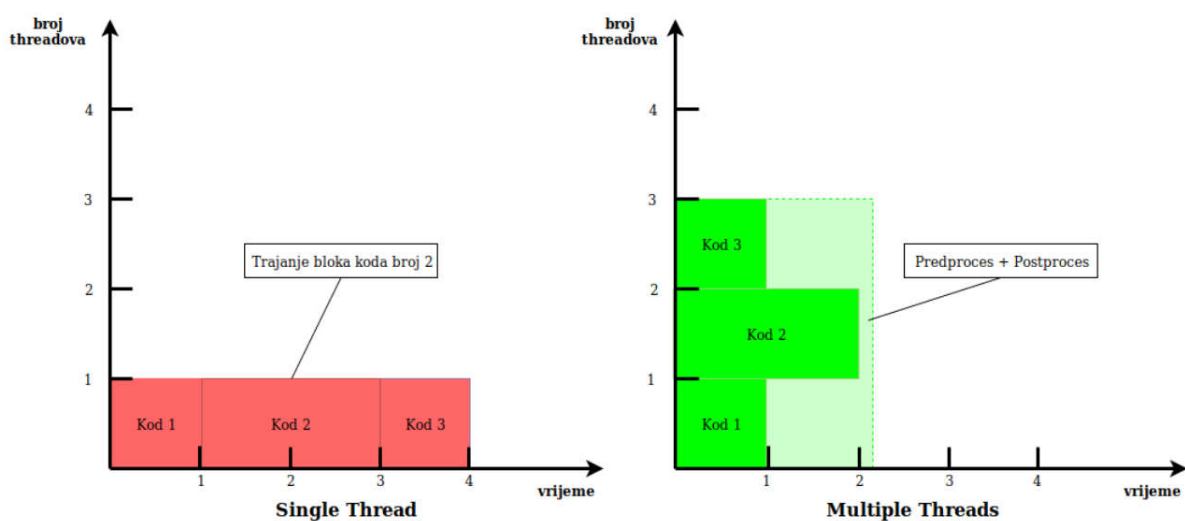
Što je zapravo *thread*? Thread je tako reći niža razina od procesa. Ne



koriste mnogo resursa, te omogućavaju da se djelovi vašeg koda pokreću neovisno jedan o drugom te tako paralelno izvršavaju kod.

Slika 24. Grafički prikaz rada threadova

Uzmimo za primjer, da imamo 3 for petlje te recimo, zbog jednostavnosti, da svaka zahtijeva 1 sekundu za izvršavanje. Ukupno izvršavanje tog programa traje 3 sekunde. Stavimo li svaku for petlju u zasebni thread, dobiva se mnogo bolji rezultat. Iz razloga što je svaka petlja u svom threadu, izvršavaju se paralelno te tada izvršavanje traje malo više od 1 sekunde (predprocesiranje + 1s + postprocesiranje).



Slika 25. Primjer optimizacije koda putem threadova

Danas postoje programski jezici koji nisu tzv. multi-threaded jezici, odnosno ti jezici rade na jednom threadu sve. Python je multi-threaded jezik, stoga za rad s threadovima sve što nam je trebalo za rad s threadovima jest uvesti paket *threading*. Navedeni paket nudi jednostavno korištenje threadovima uz uvjet poznavanja rada s njima.

- **import threading**- omogućava korištenje paketa Threading u kodu
- **threading.Thread**- klasa Thread iz koje naše klase u kodu nasljeđuju

- **threading.Thread.start()** - metoda koja stvara thread i pokreće svoju metodu run()
- **threading.Thread.run()** - metoda koja sadrži kod koji se pokreće u stvorenom threadu odmah nakon stvaranja
- **threading.Thread.close()** - metoda koja zatvara thread nad kojim je pozvana

Sada za ilustraciju našeg problema, zamislimo da dolaze dva zahtjeva s MacroToucha na korisničko računalo. Bez threadova, postoji mogućnost da se zbog obrađivanja prvih podataka, drugi podaci izgube. Uz threadove, pri dolasku zahtjeva stvara se poseban thread za njega koji se zatvara pri završetku obrade. Na taj način smo sačuvali svaki poslani podatak.

Struktura koda:

```
desktop
├── app.py
├── Pipfile
├── Pipfile.lock
├── servers
│   ├── connection.py
│   ├── __init__.py
│   ├── macro.py
│   ├── monitor_utilities.py
│   └── system_monitor.py
├── start.sh
├── UdpClient.class
└── UdpClient.java
```

Folder desktop, sa strane komunikacije, sadrži samo serverski dio aplikacije. Svi serveri se nalaze u folderu servers i pri pokretanju aplikacije oni se zapravo pokreću na korisničkom računalu.

Utilities folder sadrži sve one klase i funkcije koje mogu biti potrebne u serverima za obrađivanje dobivenih podataka ili možda dobavljanje nekakvih podataka.

Java klasa i file smo koristili kako bi optimizirali simuliranje micanja miša zbog izvrsnog paketa Robot.

Folder raspberry sadrži, sa strane komunikacije, pretežno sockete koji šalju zahtjeve na desktop. Pa tako controllers folder u sebi sadrži klase koje šalju makro naredbe ili micanje miša na desktop aplikaciju.

Creator folder u sebi sadrži sve one klase koje služe za dodavanje osobnih konfiguracija ili u budućnosti možda čak i widgeta. Te konfiguracije se dakako spremaju u json file koji se nalazi u središnjem folderu.

PyUserInput

Srž naše ideje jest bez muke simulirati pritisak makro naredbe. Kako bi to uopće funkcioniralo, potrebno je simulirati pritisak tipki na korisničkom računalu. Python, srećom, ima inuitivan paket zvan PyUserInput. On na vrlo jednostavan način omogućava simulaciju pritiska tipke. Na taj način smo riješili računalni dio softwarea.

- **import PyUserInput**- omogućava korištenje paketa koji su dio PyUserInputa; PyKeyboard, PyMouse...
- **PyKeyboard()** - stvara objekt virtualne tipkovnice
- **PyKeyboard.alt_key**- predstavlja pozicijsku vrijednost ALT tipke, analogno vrijedi i za ostale tipke
- **PyKeyboard.press_keys(key_sequence)** - metoda koja prima listu tipki te simulira pritisak njihov pritisak
- **PyKeyboard.tap_key(key)** - metoda prima integer koji predstavlja poziciju te tipke na tipkovnici i simulira njen pritisak

Naš uređaj ima već navedenu sposobnost dodavanja makro naredbi. Pri stvaranju te opcije suočili smo se sa problemom kompatibilnosti. Paket PyUserInput ima zasebne oznake specijalnih tipki za Windows, MacOS i Linux distribucije. Kako ne bi svakog puta provjeravali sistemske postavke

korisničkog računala te potom pozivali ovisno o tome određeni algoritam, našli smo drugo rješenje. Uočili smo da unatoč drugim imenima tipki (primjerice *Command Control*), tipke sličnih odnosno istih funkcija nalaze se pod istom pozicijskom vrijednosti (*Command* i *Control* nose vrijednosti 37). S tom informacijom smo kreirali objekt rječnika koji sadrži imena tipki i pretvara ih u numeričku vrijednost te tipke. Ovako to izgleda u kodu:

```
from pykeyboard import PyKeyboard
k = PyKeyboard()
SPECIAL_KEYS_DICT = {
    "CTRL": k.control_key,
    "ALT": k.alt_key,
    "F1": k.function_keys[1],
    "F2": k.function_keys[2],
    "F3": k.function_keys[3],
    "F4": k.function_keys[4],
    "F5": k.function_keys[5],
    "F6": k.function_keys[6],
    "F7": k.function_keys[7],
    "F8": k.function_keys[8],
    "F9": k.function_keys[9],
    "F10": k.function_keys[10],
    "F11": k.function_keys[11],
    "F12": k.function_keys[12],
    "TAB": k.tab_key,
    "SHIFT": k.shift_key,
    "ESCAPE": k.escape_key,
    "PRTSCR": k.print_screen_key,
    "INSERT": k.insert_key,
    "DELETE": k.delete_key,
    "PAGE_UP": k.page_up_key,
    "PAGE_DOWN": k.page_down_key,
    "HOME": k.home_key,
    "BACKSPACE": k.backspace_key,
    "SUPER": k.super_l_key,
}
```

Dakle, treba priznati da PyUserInput nije uredno riješio problem kompatibilnosti, stoga smo mi morali improvizirati. No kako nam je on ključan za nastavak razvitka ovog projekta, urediti ćemo sami paket i riješiti problem kompatibilnosti.

JAVA I JAVASCRIPT

Kad smo krenuli tek u izradu aplikacije, i kad su se radili prvi koncepti, shvatili smo da nam je brzina ključna. Python nam to u jednom slučaju nije mogao pružiti. Pythonov paket `PyUserInput` je vrlo sporo simulirao kretanje miša, i stoga smo se dali u potragu za nekim drugim načinom. Pronašli smo ga u Javi. Java ima paket zvan *Robot* koji jako brzo i optimizirano simulirano kretanje miša. Stoga smo uzeli njega kako bi aplikacija bila što optimiziranija.

Javascript smo koristili samo u jednom fileu. Javascript nam je ponudio rješenje problemu s kojim smo se suočili i to u samo nekoliko linija koda. Za rad dijela našeg softwarea potrebno je dohvatiti program koji je u fokusu na korisničkom računalu. Kako svaki OS na različiti način dohvaća te podatke, naišli smo na problem. Smatrali smo da mora postojati jednostavan univerzalan način za izvesti to. Tako smo se dali u potragu za korisnim paketima, i tu nam je rješenje ponudio Javascript. Naime postoji modul *active-window*, koji je već odradio posao kompatibilnosti i vama samo vraća prozor koji se koristi. Tu skriptu smo pozivali unutar python koda.



VIII. poglavlje

ZAHVALA



Zahvala

Na kraju se želimo zahvaliti svima koji su svo ovo vrijeme bili uz nas i pomogli nas da ovaj projekt privedemo kraju.

Među njih spadaju:

profesor Ante Bartulović zajedno sa gospodinom Damirom Brčićem koji su nam putem PICS-a pružili opremu za rad i odlično radno okruženje u kojem smo razvijali veliki dio našeg projekta.

Vanjski mentor Ivan Lukšić koji je uz nas bio od samog početka uvijek spreman pružiti bilo kakvu uslugu, koji nas je poticao na rad, hvalio sve naše uspjehe i analizirao sve probleme u kojima smo se našli.

Tibor Zaviršek, comicrelief cijelog procesa rada na projektu i osoba koja nas je naučila kako normalno držati kameru.

Svi prijatelji i kolege koji su trpili naše ispade posljednjih tjedan dana.

Mentorica Sani Dužević koja je u posljednje četiri godine od skupine individualaca stvorila tri osobe koje su spremne na sve što svijet velikih IT-evaca može baciti na njih, koja je slušala sve naše glupe ideje i dopuštala nam da sami shvatimo koliko su glupe. Osim što nam je pomogla da naučimo sve što dosada znamo, upućivala nas gdje da idemo i upozoravala na sve što nas može zadesiti na putu, pomogla nam je izgraditi karakter i vlastito ja i zbog toga smo joj neizmjereno zahvalni.

