

RAČUNARSKA GIMNAZIJA

Lazar Premović

# ČOVEK-MAŠINA INTERFEJS ZA PRIMENU U MUZICI

Professional Hardware Interfaces - Generic Professional  
Controller "PHI - GPC"

maturski rad iz predmeta:  
**Napredne tehnike programiranja**

**Mentor:**

Prof. dr Filip Marić

Beograd, maj 2019.

**Naslov maturskog rada:** Čovek-mašina interfejs za primenu u muzici

**Rezime:** Upravljanje računarom uz pomoć miša i tastature je sasvim dovojno za prosečne korisnike ali za većinu profesionalnih korisnika (pogotovo u Audio/Video oblasti) miš i tastatura nisu optimalan način upravljanja računarom. Zato se za većinu profesionalnih primena, na račuare povezuju posebno dizajnirani kontroleri koji olakšavaju i ubrzavaju zadatke koje treba obaviti. Međutim kontroleri pored visokih cena imaju još nekoliko ograničenja koji ih čine nepraktičnim: dizajnirani su za određenu vrstu posla (npr. teško je obrađivati slike na kontroleru dizajniranom za miksovanje zvuka) i često rade sa samo jednim programom i vrlo je teško ili nemoguće koristiti ih za upravljane nekim drugim programom.

Zato je cilj ovog rada izrada kontrolera koji je primenljiv u širokom spektru poslova i moguće ga je povezati sa bilo kojim programom. Da bi se taj cilj ostvario, hardver kontrolera mora biti opšte koncipiran<sup>1</sup> da bi mogao da se primeni na bilo koju vrstu softvera kojim treba da se upravlja, a softver za konfigurisanje mora da omogući veoma detaljnu konfiguraciju kontrolera i lako proširivanje funkcionalnosti.

Tako su nastali:

- **GPC "Generic Professional Controller"** (Kontroler),
- **PHI Control Center**<sup>2</sup> (Softver za konfigurisanje i obradu u realnom vremenu),
- **GHP "Generic HID<sup>3</sup> Protocol".** (Protokol za komunikaciju između kontrolera i softvera).

---

<sup>1</sup>Ne specijalizovan ni za jednu konkretnu primenu.

<sup>2</sup>Professional Hardware Interfaces Control Center.

<sup>3</sup>Human Interface Device

# Sadržaj

<b>1 Korišćene tehnologije</b>	<b>1</b>
1.1 Arduino . . . . .	1
1.2 C# desktop aplikacije . . . . .	2
1.3 JSON . . . . .	2
1.4 MIDI . . . . .	2
<b>2 GHP Protokol</b>	<b>4</b>
<b>3 Kontroler</b>	<b>6</b>
3.1 Hardver . . . . .	7
3.2 Elektronika i integrisana kola . . . . .	8
3.3 Mikrokontrolerski kod . . . . .	10
<b>4 Desktop softver</b>	<b>15</b>
4.1 Upotreba softvera . . . . .	15
4.2 Detaji implementacije . . . . .	17
<b>5 Zaključci i dalji rad</b>	<b>22</b>
<b>Dodatni materijali</b>	<b>23</b>
<b>Literatura</b>	<b>31</b>

# Glava 1

## Korišćene tehnologije

### 1.1 Arduino

Arduino je projekat otvorenog koda koji dizajnira i proizvodi mikrokontrolere bazirane na *Atmel* 8-bitnim AVR mikrokontrolerima čije se sve neophodne komponente nalaze na jednoj štampanoj ploči. Arduino mikrokontroleri se programiraju pomoću Arduino IDE-a<sup>1</sup> koristeći programski jezik C++ i nekoliko dodatnih biblioteka. U kodu je umesto *main()* funkcije neophodno imati funkcije: *setup()* koja se ivzrši samo jednom čim se mikrokontroler uključi i *loop()* koja se poziva dokle god je mikrokontroler uključen. Arduino IDE takođe omogućava prevođenje koda i njegovo učitavanje na mikrokontroler pomoću samo jednog klikna čime time olakšava i ubrzava razvoj softvera za Arduino mikrokontrolere. Digitalni pinovi mogu biti u samo dva stanja: logički 1<sup>2</sup> i logički 0<sup>3</sup> u daljem tekstu ćemo ih zvati HIGH i LOW.

**PWM** Pulse Width Modulation, je tehnika kojom je moguće aproksimirati bilo koji izlazni napon čak iako pin može biti samo u HIGH ili LOW stanju. To se postiže tako što pin vrlo brzo menja stanja a podešavanjem odnosa vremena kada je pin HIGH i LOW moguće je tačno precizirati traženi napon.

---

<sup>1</sup>Integrated Development Environment - Integrisano razvojno okruženje.

<sup>2</sup>Odgovara naponu od 5V

<sup>3</sup>Odgovara naponu od 0V

## 1.2 C# desktop aplikacije

C# je programski jezik opšte namene koji je razvio Microsoft za potrebe razvojnog okruženja **.NET Framework**. C# podržava objektno orijentisani, imperativnu, deklarativnu i generičku programsku paradigmu. Deo C#-a je i paket **Windows Forms** koji omogućava brzo i lako kreiranje desktop aplikacija sa grafičkim interfejsom uz pomoć Microsoft Visual Studio IDE-a.

## 1.3 JSON

JavaScript Object Notation **JSON** je format za čuvanje i prenos podataka u kome su podaci zabeleženi tako da ih je vrlo lako učitati i prevesti u odgovarajući objekat. Svaki zabeleženi podatak je predstavljen kao par ključa i vrednosti. Tipovi podataka koje JSON podržava su: string,int,bool,null,niz i objekat (koji može sadržati bilo koji tip).

## 1.4 MIDI

Musical Instrument Digital Interface **MIDI** je standard koji opisuje komunikacioni protokol, digitalni interfejs i fizičke konektore koji omogućuju povezivanje raznovrsnih digitalnih audio uređaja. MIDI protokol definiše MIDI poruke koje su sačinjene od statusnog bajta i do dva bajta sa podacima koji se prenose serijski brzinom od  $31.25 \text{ kbit/s}$ . Statusni bajtovi su uvek  $\geq 128$  dok su bajtovi sa podacima  $< 128$ . MIDI definiše nekoliko statusnih bajtova kao što su NoteOn (nota je pritisnuta), NoteOff (nota je otpuštena) i ControlChange (vrednost kontrole se promenila), koji je bitan zato što se uglavnom koristi za komunikaciju kontrolera i audio softvera. MIDI definiše da posle ControlChange statusnog bajta slede dva bajta: prvi bajt sadrži ID kontrole iz opsega 0-127 i drugi bajt koji sadrži novu poziciju kontrole iz opsega 0-127.

### virtualMIDI

Kako MIDI standard definiše samo komunikaciju između dva hardverska uređaja ili između hardvera i softvera. Da bi ostvarili komunikaciju između dva softvera (u ovom slučaju odabranog audio softvera i **PHI Control Center**-a koji služi kao posrednik između audio softvera i kontrolera) koristimo biblioteku koja nam

## *GLAVA 1. KORIŠĆENE TEHNOLOGIJE*

---

omogućava kreiranje virtuelnog MIDI uređaj i njegovo korišćenje za slanje MIDI poruka. virtualMIDI Tobias Erichsen-a je biblioteka koja nam to omogućava. Za njenu korišćenje potrebno je instalirati loopMIDI ili rtpMIDI softver, uz koji dolazi državnik za koji se virtualMIDI povezuje. Nakon toga je dovoljno uključiti .cs fajl biblioteke u projekat i koristiti metode za kreiranje virtuelnog uređaja i slanje poruka preko njega.

# Glava 2

## GHP Protokol

Da bi kontroler mogao da komunicira sa softverom za konfiguraciju osmišljen je i definisan protokol koji će se koristiti. GHP kao svoju osnovu koristi serijsku komunikaciju preko USB-a sa podešenim baud rate-om<sup>1</sup> od 115200<sup>2</sup>. Poruke koje se šalju ovim protokolom imaju sledeći format:

*CommandByte, ChannelByte, [0 – 20]DataBytes.*

**Komandni bajt** govori o kojoj komandi je reč i takođe time precizira broj bajtova sa podacima koji će uslediti (za spisak komandi pogledati tabelu 2.1). Radi lakšeg razumevanja sve vrednosti komandnih bajtova su iz alfanumeričkog opsega ASCII enkodiranja tako da ih možemo lakše obeležavati po ASCII karakteru koji im odgovara.

**Bajt koji označava kanal** govori na koju kontrolu ili memoriju adresu se odnosi komanda.

**Bajtovi sa podacima** sadrže podatke karakteristične za određenu komandu.

---

<sup>1</sup>Brzina prenosa podataka, u digitalnoj komunikaciji ekvivalentna sa brojem bitova u sekundi.

<sup>2</sup>115.2 kbit\ s.

## GLAVA 2. GHP PROTOKOL

Komandni bajt	Komanda	Kanal	Opis bajtova sa podacima
'B' 66	Taster je pritisnut	Indeks tastera	NEMA
'R' 82	Taster je otpušten	Indeks tastera	NEMA
'A' 65	Analogna kontrola je pomerena	Indeks analogne kontrole	2 bajta. Nova vrednost analogne kontrole, Little Endian enkodiranje
'I' 73	Rotacioni enkoder je inkrementovan	Indeks enkodera	NEMA
'D' 68	Rotacioni enkoder je dekrementovan	Indeks enkodera	NEMA
'L' 76	Podesi LED	Indeks diode	3 bajta. R, G, B
'l' 108	Podesi LCD	Broj linije ekrana	20 bajtova. Tekst
'P' 80	Podesi trajni podatak	Memorijska adresa	1 bajt. Vrednost
'p' 112	Zahtev za vrednost trajnog podatka	Memorijska adresa	NEMA
'r' 114	Odgovor na zahtevanu vrednost	Memorijska adresa	1 bajt. Vrednost
'S' 83	Sistemska komanda (videti tabelu 2.2)	Vrsta sistemske komande	Zavisi od vrste sistemske komande

Tabela 2.1: Objasnjenje komandnih bitova

Vrsta sistemske komande	Komanda	Opis bajtova sa podacima
'W' 87	Zahtev za identifikacioni broj uređaja	NEMA
'w' 119	Odgovor na zahtev za identifikacioni broj uređaja	1 bajt. Identifikacioni broj uređaja
'R' 82	Resetuj uređaj	NEMA

Tabela 2.2: Objasnjenje sistemskih komandi

# Glava 3

## Kontroler



Slika 3.1: GPC kontroler

Funkcija kontrolera je da akcije korisnika prenese softveru za obradu u realnom vremenu. Taj proces zahteva sledeće korake:

1. učitavanje stanja kontrola,
2. detekciju promena stanja kontrola,

## *GLAVA 3. KONTROLER*

---

3. slanje informacija o promeni stanja kontrola koristeći GHP protokol,
4. ažuriranje stanja displeja i dioda na osnovu informacija primljenih GHP protokolom.

Zato GPC ima nekoliko integralnih kola koja pomažu pri učitavanju stanja kontrola i Arduino Mega mikrokontroler koji vrši sve ostale operacije i implementira GHP protokol.

### **Karakteristike Arduino Mega mikrokontrolera**

- ATmega2560 mikrokontroler
- 54 Digitalna ulazno/izlazna pina (od kojih 15 imaju podršku za PWM<sup>1</sup>)
- 16 Analognih ulaznih pinova
- 256 KB Programske memorije
- 8 KB RAM memorije
- 4 KB EEPROM<sup>2</sup> stalne memorije

## **3.1 Hardver**

### **Kontrolna površina**

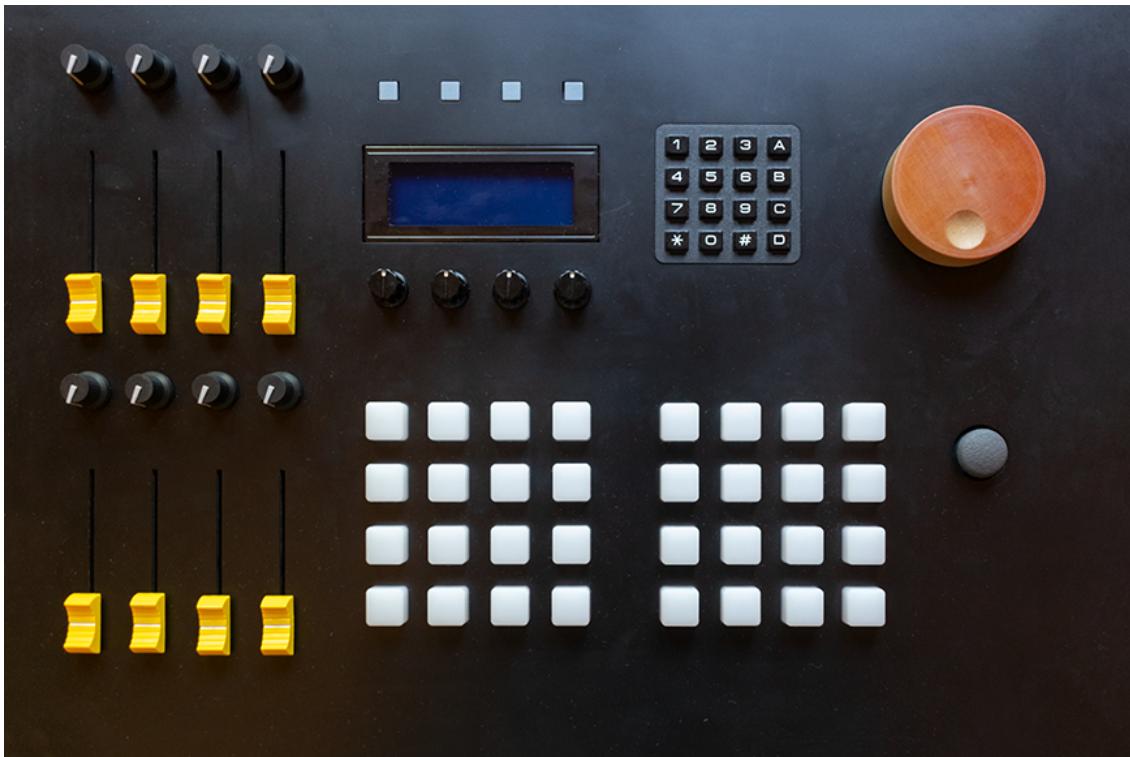
Kontrolnu površinu čine nekoliko disjunktnih grupa (videti sliku 3.2).

- Analogna sekcija (krajnje levo) sadrži 8 kliznih i 8 rotacionih potenciometara koji u zavisnosti od svoje pozicije mogu imati vrednosti u opsegu 0-1023
- Sekcija sa tasterima (dve matrice sa belim tasterima i numerička tastatura) sadrži 48 tastera podeljena u tri 4x4 matrice (donja leva matrica takođe sadrži RGB LED diode ispod svakog tastera) i šalje informaciju kada se neki od tastera pritisne ili otpusti.

---

<sup>1</sup>Pogledati 1.1.

<sup>2</sup>Electrically Erasable Programmable Read-Only Memory.



Slika 3.2: Kontrolna površina

- Displej i prateće kontole, ispod displeja se nalaze 4 beskonačna rotaciona enkodera koji šalju informaciju kada su inkrementovani<sup>3</sup> ili dekrementovani<sup>4</sup> a iznad se nalaze 4 tastera i svi su namenjeni da izvršavaju funkcije prikazane na displeju.
- Točak za brzu navigaciju (Jog wheel) i džojstik su zasebni elementi. Točak se realizuje pomoću rotacionog enkodera, a džojstik kao spoj dve analogne kontrole za x i y ose i tastera koji reaguje kada se džojstik pritisne.

## 3.2 Elektronika i integrisana kola

Da bi se gore navedene kontrole povezale potrebno je:

$$16 + 48 + 3 * 16^5 + 4 + 2 * 5^6 + 6^7 + 3 = 135$$

---

<sup>3</sup>Okrenuti za jedan stepen u smeru kazaljke na satu.

<sup>4</sup>Okrenuti za jedan stepen u smeru suprotnom od kazaljke na satu.

<sup>5</sup>Po tri za svaku LED diodu.

<sup>6</sup>Po dva za svaki enkoder

<sup>7</sup>Za displej.

### *GLAVA 3. KONTROLER*

---

pinova, mnogo više nego što mikrokontroler ima, zato je neophodno upotrebiti nekoliko tehnika kojima se taj broj smanjuje na manje od  $\approx 70$ .

**Matrice tastera** Matrice tastera nam omogućavaju da  $N$  tastera povežemo koristeći samo  $2\sqrt{N}$  pinova tako što tastere rasporedimo u matricu. Čitanje vršimo tako što po jednu kolonu postavljamo HIGH i onda čitamo vrednosti redova. Ono što poboljšava ovo rešenje je da sve ulazne<sup>8</sup> pinove svih matrica spojimo i efektivno dobijemo jednu veliku  $\sqrt{N} \times 3\sqrt{N}$  matricu koja koristi samo  $4\sqrt{N}$  pinova za  $3N$  tastera. Sličan princip može se primeniti i na LED diode, time se dobijaju tri matrice (po jedna za svaku boju) koje se opet mogu spojiti u  $\sqrt{N} \times 3\sqrt{N}$  matricu. Ovim je broj potrebnih pinova smanjen na: na:

$$16 + 16^9 + 16^{10} + 4 + 2 * 5 + 6 + 3 = 71$$

što je još uvek više od broja slobodnih pinova na mikrokontroleru.

**Multipleksiranje** Multipleksiranje nam omogućava da korsiteći specijalno integrисано kolo smanjimo broj potrebnih pinova sa  $N$  na  $(\log_2 N) + 1$ . To postižemo tako što svaki ulaz adresiramo po binarnoj reprezentaciji njegovog indeksa a broj bitova u binarnoj reprezentaciji nekog broja je  $\log_2 N$  i još jedan pin za čitanje izabranog ulaza čime dobijamo  $(\log_2 N) + 1$ . U ovom projektu su korišćena dva tipa multipleksera: multiplekser sa 8 ulaza (koristi samo 4 pina) i multiplekser sa 4 ulaza (koristi samo 3 pina). Sledeća optimizacija koju možemo da izvršimo jeste da pinove za selekciju svih multipleksera istog tipa povežemo zajedno i time dobijamo  $N * M^{11}$  ulaza sa samo  $(\log_2 N) + M$  pinova. U kontroleru se nalaze 4 multipleksera sa 8 ulaza i 3 multipleksera sa 4 ulaza. Dva multipleksera sa 8 ulaza se koriste za ulaze iz analogne sekcije dok druga dva pokrivaju izlaze iz matrice tastera i tastere iznad displeja. Svaki od multipleksera sa 4 kanala je zadužen za ulaze po jedne matrice

---

<sup>8</sup>pinove na kolonama.

<sup>9</sup> $\sqrt{N} \times 3\sqrt{N}$  matrica tastera.

<sup>10</sup> $\sqrt{N} \times 3\sqrt{N}$  matrica LED dioda.

<sup>11</sup>N broj ulaza multipleksera, M broj multipleksera.

LED dioda (po jedna matrica za svaku boju). Trenutni broj potrebnih pinova je:

$$3^{12} + 4^{13} + 2^{14} + 3^{15} + 4^{16} + 4^{17} + 2 * 5 + 6 + 3 = 39$$

što je dovoljno mali broj da se poveže na mikrokontroler.

Za šemu elektronike pogledajte sliku 1, a detaljnije šeme i tehnički crteži se nalaze na priloženom CD-u.

### 3.3 Mikrokontrolerski kod

Kod koji se izvršava na mikrokontroleru se sastoji od glavnog fajla koji sadrži funkcije *setup()* i *loop()* i još nekoliko biblioteka koje sam napisao<sup>18</sup>, a to su:

- **define** biblioteka koja sadrži sve konstante (npr. brojeve pinova) i globalne promenljive koje su potrebne,
- **lib** biblioteka koja sadrži neke funkcije koje mogu biti korisne i u drugim projektima (upravljanje matricama tastera, multiplekserima i opreacije nad nizovima pinova),
- **GHP** biblioteka koja implementira GHP protokol za komunikaciju sa računaram,
- **Persistence** biblioteka koja olakšava trajno skladištenje i čitanje podataka.

### Čitanje tastera

Ukoliko su tasteri lošeg kvaliteta ili se ne pritisnu u potpunosti, može doći do odskakivanja kontakata unutar tastera što se može registrovati kao više pritisaka umesto samo jednog. Zato je na svim tasterima kontrolera implementiran *Debouncing*<sup>19</sup> algoritam.

---

<sup>12</sup>Selektori za multipleksere sa 8 kanala.

<sup>13</sup>Ulazi za multipleksere sa 8 kanala.

<sup>14</sup>Selektori za multipleksere sa 4 kanala.

<sup>15</sup>Izlazi za multipleksere sa 4 kanala.

<sup>16</sup>Izlazi za matricu tastera.

<sup>17</sup>Izlazi za matricu LED dioda.

<sup>18</sup>Sve biblioteke su sačinjene od istoimenih .cpp i .h fajlova.

<sup>19</sup>Algoritam za uklanjanje odskakivanja.

```
byte val=digitalRead(MUXDR[0]);
if(val==HIGH && ButtonDebounce[ind]<DEBOUNCE)
{
    ButtonDebounce[ind]++;
    if(ButtonDebounce[ind]==DEBOUNCE && ButtonBool[ind]==false)
    {
        SendMessage(BUTTON_PRESS,ButtonAlias[ind]);
        ButtonBool[ind]=true;
    }
}
else if(val==LOW && ButtonDebounce[ind]>0)
{
    ButtonDebounce[ind]--;
    if(ButtonDebounce[ind]==0 && ButtonBool[ind]==true)
    {
        SendMessage(BUTTON_RELEASE,ButtonAlias[ind]);
        ButtonBool[ind]=false;
    }
}
```

Algoritam radi tako što za svako dugme pamti broj koji se inkrementuje ako je dugme pritisnuto a dekrementuje ako nije i proverava se da li je ta vrednost jednaka nuli (što znači da je dugme već neko vreme sigurno otpušteno) ili je ta vrednost jednaka DEBOUNCE konstanti (što znači da je dugme već neko vreme pritisnuto) i ako se novo stanje dugmeta razlikuje od poslednjeg stanja koje smo poslali računaru šaljemo mu novo stanje. DEBOUNCE konstanta je podesiva sistemom trajnih podataka i govori programu koliko dugo je potrebno da dugme bude pritisnuto ili otpušteno pre nego što se kao takvo i registruje.

## Čitanje analognih kontrola

Analogne kontrole su takođe podložne unošenju šuma ali za njih je potrebno koristiti drugačije algoritme, nekada je čak potrebno da neke kontrole imaju drugačije algoritme za otklanjanje šuma od ostalih. GPC na većini kontrola implementira kombinaciju dva algoritma.

```

short val = analogRead(MUXAR[0]);
val = constrain(map(val,0,1005,0,1023),0,1023);
short avg = (val+AnalogAvg[i])/2;
AnalogAvg[i]=val;
if(abs(avg-AnalogPre[i])>TRESHOLD)
{
    SendMessage(ANALOG_CHANGE,AnalogAlias[i],avg,true);
    AnalogPre[i]=avg;
}

```

Prvo se pre bilo kakvog filtriranja vrši mapiranje vrednosti čime se osigurava da je moguće dostići maksimalnu vrednost koja je 1023. Potom se kao prvi korak u otklanjanju šuma koristi prosečna vrednost poslednja dva čitanja a informacija se šalje računaru samo ako se razlikuje za više od konstante TRESHOLD od poslednje poslate vrednosti čime se dodatno smanjuje šum i protok nepotrebnih informacija. Specijalni slučaj je analogna pedala na opcionom dodatku za ovaj kontroler, zato što pedala ima izuzetno mali hod (rezolucije samo 50 jedinica) gore navedeni algoritam nije dobro rešenje zbok gubitka rezolucije koji je njemu svojstven. Stoga je na pedali implementiran **Eksponencijalni rekurzivni filter** koji se implementira preko formule:

$$R_i = V * E + R_{i-1} * (1 - E)$$

gde je  $R$  izlaz filtera,  $V$  pročitana vrednost kontrole i  $E$  konstanta filtera u opsegu 0-1.

## Rotacioni enkoderi

Rotacioni enkoderi rade na principu dva signala koji su  $90^\circ$  van faze tako da je moguće detektovati pomeraj enkodera tražeci rastuću<sup>20</sup> ili opadajuću<sup>21</sup> ivicu na jednom signalu a onda odrediti smer rotacije na osnovu drugog signala. Zbok preciznosti je neophodno detektovati svaku rastuću ivicu. I zato ručno skeniranje u svakoj iteraciji *loop()* funkcije nije dobro rešenje jer može preskočiti neku ivicu ukoliko se enkoder vrti brzo i procesor je opterećen. Dobro rešenje za detekciju rastućih ivica koristi sistemske prekide čiju upotrebu je Arduino pojednostavio. Za početak potrebno je omogućiti sistemske prekide na nekom pinu i proslediti mu funkciju koja se izvršava kada se prekid desi, a to se postiže sledećom linijom koda: `attachInterrupt(digitalPinToInterrupt(ENC[0][0]), ENCOISR, RISING);` gde je ENCOISR funkcija koju će prekid pozvati.

<sup>20</sup>Prelaz pina sa LOW na HIGH stanje.

<sup>21</sup>Prelaz pina sa HIGH na LOW stanje.

### GLAVA 3. KONTROLER

---

```
void ENCOISR()
{
    if(digitalRead(ENC[0][0])==HIGH)
    {
        if(digitalRead(ENC[0][1])==LOW)
        {
            ENCPre[0]++;
        }
        else
        {
            ENCPre[0]--;
        }
    }
}
```

Sistemski prekidi su veoma osetljivi na odskakivanje kontakata<sup>22</sup> a enkoderi su veoma poldožni odskakivanju i zato je neophodno imati dosta filtriranja pri radu sa njima. Prvi korak je da u samoj funkciji proverimo da li je pin zaista HIGH i ako jeste onda ažuriramo relativnu poziciju enkodera u odnosu na prošlu iteraciju.

```
void Enc()
{
    for(byte i=0;i<5;i++)
    {
        ENCPre[i]=constrain(ENCPre[i],-((short)ENCMS),(short)ENCMS);
        while(ENCPre[i]!=0)
        {
            if(ENCPre[i]>0)
            {
                if(ENClast[i]==true)
                {
                    SendMessage(ENCODER_INCREMENT,i);
                }
                ENCPre[i]--;
                ENClast[i]=true;
            }
            else{
                if(ENClast[i]==false)
                {
                    SendMessage(ENCODER_DECREMENT,i);
                }
                ENCPre[i]++;
                ENClast[i]=false;
            }
        }
    }
}
```

---

<sup>22</sup>Videti 3.3.

### *GLAVA 3. KONTROLER*

---

Funkcija *Enc()* se izvršava svake iteracije *loop()* funkcije i šalje najnovije podatke u vezi sa enkoderima računaru. Funkcija prolazi kroz svaki enkoder i prvo ograniči broj pomeraja na konstantu **ENCMS** koja označava maksimalni broj koraka enkodera koji će se registrirati za vreme jedne iteracije. Nakon toga procesiramo promenu relativne pozicije i ukoliko je ista kao i predhodna obaveštavamo kompjuter o njoj, ovo je još jedan vid filtriranja koji sprečava da se u nizu operacija inkrementovanja jedna pogrešno protumači kao dekrementovanje i obratno.

# Glava 4

## Desktop softver

Desktop softver je drugi bitan deo GHP ekosistema i služi da omogući konfiguraciju kontrolera i da u realnom vremenu prevodi komande sa kontrolera u nešto što ciljani softver razume.

### 4.1 Upotreba softvera

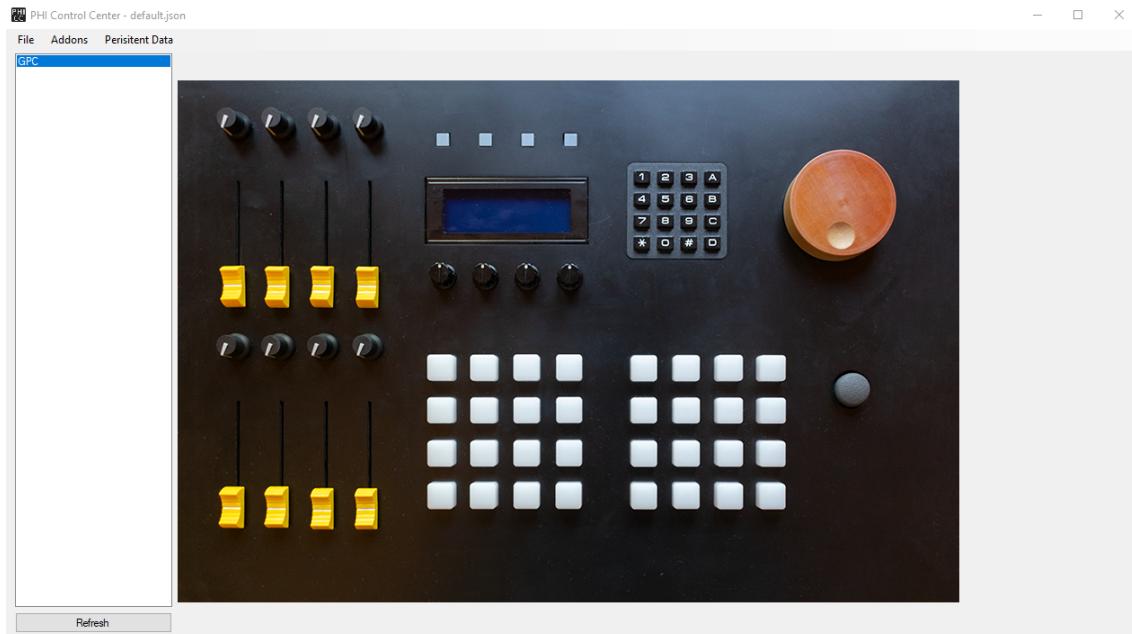
Pokretanjem softvera pojavljuje se ikonica u donjem desnom uglu trake sa zadatacima. Klikom na ikonicu pojavljuje se meni koji omogućava lak pristup glavnim segmentima softvera.

#### Segment za konfigurisanje

Na početnom ekranu segmenta za konfigursanje se nalaze:

- Lista svih trenutno povezanih uređaja,
- Dugme koje osvežava listu svih povezanih uređaja,
- Vizuelna reprezentacija trenutno izabranog uređaja na kojoj se i vrši samo konfigurisanje uređaja,
- File meni koji omogućava učitavanje, čuvanje i biranje aktivnog profila,
- Meni za aktiviranje i deaktiviranje opcionih dodataka,
- Meni za pregled i izmenu trajnih podataka na kontroleru.

## GLAVA 4. DESKTOP SOFTVER



Slika 4.1: Početni ekran segmenta za konfigurisanje

Konfigurisanje određene kontrole se postiže klikom na nju na vizuelnoj reprezentaciji kontrolera, čime se pojavljuje dijalog u kome je moguće izabrati funkciju koju želimo da ta kontrola obavlja a klikom na taster ... moguće je podesiti dodatne parametre funkcije<sup>1</sup>.

Dijalog za upravljanje dopunskim kontrolerima sadrži listu svih dopunskih kontro-



Slika 4.2: Izbor funkcije za kontrolu

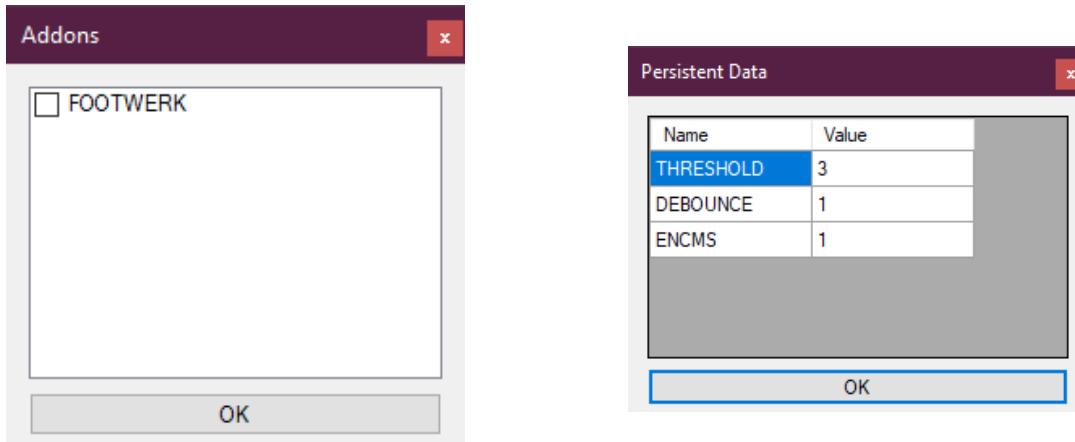
Slika 4.3: Podešavanje dodatnih parametara

lera koji se mogu povezati na izabrani kontroler i svaki od njih se može pojedinačno aktivirati ili deaktivirati klikom na kvadratić pored njegovog imena. Promene je neophodno sačuvati pritiskom na taster **OK**.

<sup>1</sup>U većini slučajeva je neophodno podesiti dodatne parametre da bi kontrola funkcionišala kako želimo.

## *GLAVA 4. DESKTOP SOFTVER*

Podešavanje trajnih podataka se vrši u svom dijalogu gde se za trenutno selektovani kontroler izlistavaju svi trajni podaci sa njihovim trenutnim vrednostima. Vrednosti je moguće promeniti direknim unosom u polje u kome se nalazi vrednost tog podatka, pritiskom na taster **OK** se sve promene vrednosti ažurirati na kontroleru.



Slika 4.4: Upravljanje opcionim dodatcima

Slika 4.5: Podešavanje trajnih podataka

### **Segment za obradu u realnom vremenu**

Segment za obradu u realnom vremenu služi da informacije o stanju kontrola koje primi od kontrolera prevede u komande koje je korisnik definisao u trenutno aktivnom profilu<sup>2</sup>. Na korisniku je samo da startuje nit koja vrši obradu u realnom vremenu klikom na dugme **Start RTP**. Bitno je napomenuti da i segment za konfiguraciju i segment za obradu u realnom vremenu komuniciraju sa povezanim uređajima. Stoga nije moguće da oba budu aktivna istovremeno pa je potrebno isključiti segment za obradu u realnom vremenu pri promeni konfiguracije uređaja.

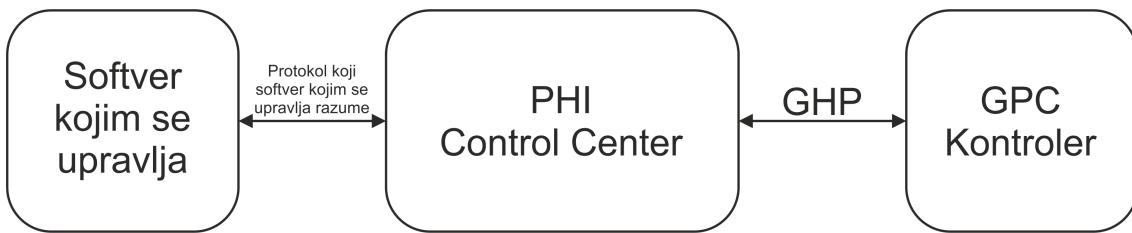
## **4.2 Detali implementacije**

### **Osnovni protokol informacija i dodatne komponente**

Osnovni dijagram protoka informacija izgleda ovako (slika 4.6). Radi lakšeg održavanja softvera i proširivanja njegovih mogućnosti, softver je podeljen na manje

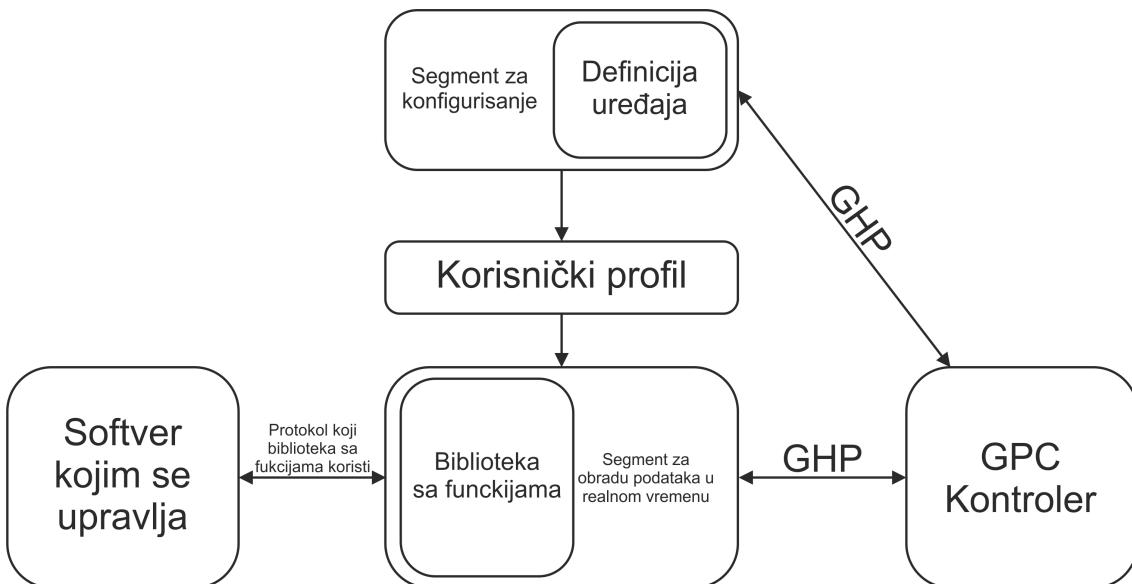
<sup>2</sup>Trenutno aktivni profil je uvek fajl `default.json` u `Profile` direktorijumu.

## GLAVA 4. DESKTOP SOFTVER



Slika 4.6: Osnovni dijagram protoka informacija

zasebne delove, čime je dobijen detaljniji dijagram protoka informacija (vidi sliku 4.7). Novi elementi koje smo uveli su:



Slika 4.7: Detaljniji dijagram protoka informacija

- **Definicija uređaja** je .json fajl koji uz sliku samog kontrolera opisuje kontroler koji se može povezati. Za svaki uređaj koji je moguće povezati potrebno je da u folderu `Devices` postoji njegova definicija uređaja i slika samog kontrolera koja se koristi za njegovu vizuelnu reprezentaciju. U samoj definiciji uređaja se pamte: ime uređaja, njegov identifikacioni broj, imena i adrese svih trajnih podataka na uređaju, imena i adrese svih opcionih dodataka i podaci o svakoj kontroli na uređaju. Za svaku kontrolu se beleži njen tip, index, pozicija na vizuelnoj reprezentaciji (kako bi bilo moguće kliknuti na nju) i još neki parametri u vezi sa njom<sup>3</sup>, sve kontrole su takođe organizovane u logičke

<sup>3</sup>Npr. za displej je zapisan broj kolona i redova na njemu.

celine. Primer jedne definicije uređaja možete videti u dodatnim materijalima 1.

- **Korisnički profil** je .json fajl koji opisuje mapiranje svake kontrole na neku određenu funkciju neke biblioteke sa funkcijama i njene dodatne parametre. Kontrole se grupišu po svom tipu pa se unutar toga adresiraju po svom indeksu i za svaku kontrolu se beleži kojoj grupi pripada. Jedno mapiranje se sastoji iz imena biblioteke kojoj funkcija pripada, imena same funkcije i niza dodatnih parametara.
- **Biblioteka sa funkcijama** je .dll fajl i možemo ih imati koliko god želimo u Plugins folderu. Sve one se učitavaju pri pokretanju softvera i moguće ih je koristiti u konfiguracijama. Svaka biblioteka mora da implementira interfejs IPlugin koji izgleda ovako:

```
public interface IPlugin
{
    Dictionary<string, FunctionBox> GetFunctions();
    bool Selftest();
    string GetPluginName();
    void Init();
    void Close();
}
public class FunctionBox
{
    public delegate void Function(byte chan, int val, List<object> par);
    public Function function;
    public CType functionType;
    public IDialogForm dialogForm;
    public FunctionBox(Function f, CType fType, IDialogForm df)
    {
        function = f;
        functionType = fType;
        dialogForm = df;
    }
    public List<object> aditionalArgs(List <object> start)
    {
        return dialogForm.go(start);
    }
}
public enum CType
{
    Analog = 0,
    Button = 1,
    RGBButton = 2,
    Encoder = 3,
    LCD = 4,
```

```
    JoyStick = 5,
    RGBLED = 6
}
public interface IDialogForm
{
    List<object> go(List<object> start);
}
```

## Detalji implementacije GHP protokola

Implementacija GHP protokola se nalazi u klasi GHP čija jedna instanca predstavlja komunikaciju sa jednim GHP uređajem<sup>4</sup> na jednom serijskom portu, tako da će nit za obradu u realnom vremenu imati po jednu instancu ove klase za svaki povezani kontroler. GHP klasa takođe ima i statičke metode koje se koriste za detekciju i inicijalno povezivanje sa uređajima kao i statičke metode koje segment za konfiguraciju koristi jer segment za konfiguraciju za razliku od segmenta za obradu u realnom vremenu ne održava stalnu konekciju sa uređajima. Sve statičke funkcije ili koriste ili se u nekoj meri baziraju na `DeviceConnectedToPort` funkciji koja za zadati serijski port vraća identifikacioni broj uređaja koji je na njega povezan ili 0 ukoliko ni jedan uređaj nije povezan na taj port. Njena implementacija izgleda ovako:

```
public static byte DeviceConnectedToPort(string com)
{
    SerialPort tmp = new SerialPort();
    tmp.BaudRate = 115200;
    tmp.PortName = com;
    tmp.ReadTimeout = 500;
    tmp.Open();
    tmp.DiscardInBuffer();
    tmp.Write(new byte[] { SYSTEM, SYSTEM_DISCOVER }, 0, 2);
    try
    {
        int Bc = 0;
        while (tmp.ReadByte() != RESPONCE_DISOVER && Bc < 75) { Bc++; }
        if (Bc == 75)
        {
            tmp.Close();
            return 0;
        }
        byte res = (byte)tmp.ReadByte();
        tmp.Close();
        return res;
    }
}
```

---

<sup>4</sup>Uredaj koji implementira GHP protokol kao što je GPC kontroler.

```
    }
    catch
    {
        tmp.Close();
        return 0;
    }
}
```

Funkcija prvo otvori port i pošalje poruku za identifikaciju uređaja i nakon toga čita bajtove sa porta dok ne pročita sve bajtove (i time istekne timeout operacije čitanja) ili pročita 75 bajtova i nijedan od njih nije odgovor na zahtev za identifikaciju (u oba slučaja funkcija vraća nulu što znači da uređaj nije pronađen) ili ukoliko u tih 75 bajtova pronađe odgovor vraća identifikaciju uređaja. Ostatak GHP klase nije preterano različi od GHP implementacije na samom kontroleru.

### **Detalji implementacije niti za obradu podataka u realnom vremenu**

Nit za obradu podataka sama po sebi nije preterano komplikovana. Pri startovanju učitava se aktivni profil (`default.json`), učitavaju se sve biblioteke sa funkcijama i proverava se za svaki serijski port da li na njemu postoji uređaj i ukoliko postoji u listu se dodaje nova instanca GHP klase vezana za taj port. Dok se niti izvršavaju u svakoj iteraciji se za svaku instancu GHP klase poziva funkcija koja obrađuje sve poruke koje su stigle u međuvremenu.

# Glava 5

## Zaključci i dalji rad

Koristeći softverska i hardverska rešenja prikazana u ovom radu moguće je vrlo lako koristiti jedan kontroler za široki spektar primena i upravljanje najrazličitijim aplikacijama. Takođe su prikazana i rešenja za dizajniranje softvera i hardvera kompatibilnog sa GHP ekosistemom. Kontroleri koji koriste GPH protokol ne moraju biti dizajnirani sa određenim softverom u vidu. A softveri mogu da implementiraju svoju biblioteku funkcija i da odmah omoguće korisnicima da koriste veliki broj različitih kontrolera da upravljaju njima.

Softver je struktuiran tako da naknadno proširivanje funkcionalnosti bude vrlo lako zahvaljujući bibliotekama sa funkcijama i definicijama uređaja. Ažuriranje samog softvera ne bi trebalo da bude neophodno osim malih promena koje bi omogućile neku dodatnu funkcionalnost bibliotekama sa funkcijama. Potenjalna unapređenja softvera bi mogla biti:

- Podrška za enkodere, LED diode i Lcd displej u biblioteci MIDI funkcija,
- Biblioteka koja omogućava da kontrole emuliraju miš i tastaturu,
- Biblioteka koja omogućava integraciju sa *AutoHotkey* skriptama,
- Podrška za modifikacione tastere i stranice kontrola (Zahteva ažuriranje celog softvera).

# Dodatni materijali

Primer koda 1: Primer definicije uređaja

```
{  
    "Name": "GPC",  
    "DeviceId": 53,  
    "IsAddon": false,  
    "Banks":  
    [  
        {  
            "Name": "AnalogBank",  
            "Controls":  
            [  
                {"Type": 0, "Id": 0, "X": 45, "Y": 351, "W": 35, "H": 150},  
                {"Type": 0, "Id": 1, "X": 95, "Y": 351, "W": 35, "H": 150},  
                {"Type": 0, "Id": 2, "X": 145, "Y": 351, "W": 35, "H": 150},  
                {"Type": 0, "Id": 3, "X": 195, "Y": 351, "W": 35, "H": 150},  
                {"Type": 0, "Id": 4, "X": 36, "Y": 276, "W": 50, "H": 45},  
                {"Type": 0, "Id": 5, "X": 89, "Y": 276, "W": 50, "H": 45},  
                {"Type": 0, "Id": 6, "X": 142, "Y": 276, "W": 50, "H": 45},  
                {"Type": 0, "Id": 7, "X": 195, "Y": 276, "W": 50, "H": 45},  
                {"Type": 0, "Id": 8, "X": 45, "Y": 107, "W": 35, "H": 150},  
                {"Type": 0, "Id": 9, "X": 95, "Y": 107, "W": 35, "H": 150},  
                {"Type": 0, "Id": 10, "X": 145, "Y": 107, "W": 35, "H": 150},  
                {"Type": 0, "Id": 11, "X": 195, "Y": 107, "W": 35, "H": 150},  
                {"Type": 0, "Id": 12, "X": 36, "Y": 25, "W": 50, "H": 45},  
                {"Type": 0, "Id": 13, "X": 89, "Y": 25, "W": 50, "H": 45},  
                {"Type": 0, "Id": 14, "X": 142, "Y": 25, "W": 50, "H": 45},  
                {"Type": 0, "Id": 15, "X": 195, "Y": 25, "W": 50, "H": 45}  
            ]  
        },  
        {  
            "Name": "ButtonBank",  
            "Controls":  
            [  
                {"Type": 2, "Id": 0, "IdL": 0, "X": 278, "Y": 304, "W": 32, "H": 32},  
                {"Type": 2, "Id": 1, "IdL": 1, "X": 325, "Y": 304, "W": 32, "H": 32},  
                {"Type": 2, "Id": 2, "IdL": 2, "X": 372, "Y": 304, "W": 32, "H": 32},  
                {"Type": 2, "Id": 3, "IdL": 3, "X": 419, "Y": 304, "W": 32, "H": 32},  
                {"Type": 2, "Id": 4, "IdL": 4, "X": 278, "Y": 351, "W": 32, "H": 32},  
            ]  
        }  
    ]  
}
```

```

        {"Type": 2, "Id": 5, "IdL": 5, "X": 325, "Y": 351, "W": 32, "H": 32},
        {"Type": 2, "Id": 6, "IdL": 6, "X": 375, "Y": 351, "W": 32, "H": 32},
        {"Type": 2, "Id": 7, "IdL": 7, "X": 419, "Y": 351, "W": 32, "H": 32},
        {"Type": 2, "Id": 8, "IdL": 8, "X": 278, "Y": 398, "W": 32, "H": 32},
        {"Type": 2, "Id": 9, "IdL": 9, "X": 325, "Y": 398, "W": 32, "H": 32},
        {"Type": 2, "Id": 10, "IdL": 10, "X": 372, "Y": 398, "W": 32, "H": 32},
        {"Type": 2, "Id": 11, "IdL": 11, "X": 419, "Y": 398, "W": 32, "H": 32},
        {"Type": 2, "Id": 12, "IdL": 12, "X": 278, "Y": 445, "W": 32, "H": 32},
        {"Type": 2, "Id": 13, "IdL": 13, "X": 325, "Y": 445, "W": 32, "H": 32},
        {"Type": 2, "Id": 14, "IdL": 14, "X": 372, "Y": 445, "W": 32, "H": 32},
        {"Type": 2, "Id": 15, "IdL": 15, "X": 419, "Y": 445, "W": 32, "H": 32},
        {"Type": 1, "Id": 16, "X": 502, "Y": 304, "W": 32, "H": 32},
        {"Type": 1, "Id": 17, "X": 549, "Y": 304, "W": 32, "H": 32},
        {"Type": 1, "Id": 18, "X": 596, "Y": 304, "W": 32, "H": 32},
        {"Type": 1, "Id": 19, "X": 643, "Y": 304, "W": 32, "H": 32},
        {"Type": 1, "Id": 20, "X": 502, "Y": 351, "W": 32, "H": 32},
        {"Type": 1, "Id": 21, "X": 549, "Y": 351, "W": 32, "H": 32},
        {"Type": 1, "Id": 22, "X": 596, "Y": 351, "W": 32, "H": 32},
        {"Type": 1, "Id": 23, "X": 643, "Y": 351, "W": 32, "H": 32},
        {"Type": 1, "Id": 24, "X": 502, "Y": 398, "W": 32, "H": 32},
        {"Type": 1, "Id": 25, "X": 549, "Y": 398, "W": 32, "H": 32},
        {"Type": 1, "Id": 26, "X": 596, "Y": 398, "W": 32, "H": 32},
        {"Type": 1, "Id": 27, "X": 643, "Y": 398, "W": 32, "H": 32},
        {"Type": 1, "Id": 28, "X": 502, "Y": 445, "W": 32, "H": 32},
        {"Type": 1, "Id": 29, "X": 549, "Y": 445, "W": 32, "H": 32},
        {"Type": 1, "Id": 30, "X": 596, "Y": 445, "W": 32, "H": 32},
        {"Type": 1, "Id": 31, "X": 643, "Y": 445, "W": 32, "H": 32},
        {"Type": 1, "Id": 32, "X": 507, "Y": 99, "W": 20, "H": 20},
        {"Type": 1, "Id": 33, "X": 533, "Y": 99, "W": 20, "H": 20},
        {"Type": 1, "Id": 34, "X": 559, "Y": 99, "W": 20, "H": 20},
        {"Type": 1, "Id": 35, "X": 585, "Y": 99, "W": 20, "H": 20},
        {"Type": 1, "Id": 36, "X": 507, "Y": 124, "W": 20, "H": 20},
        {"Type": 1, "Id": 37, "X": 533, "Y": 124, "W": 20, "H": 20},
        {"Type": 1, "Id": 38, "X": 559, "Y": 124, "W": 20, "H": 20},
        {"Type": 1, "Id": 39, "X": 585, "Y": 124, "W": 20, "H": 20},
        {"Type": 1, "Id": 40, "X": 507, "Y": 149, "W": 20, "H": 20},
        {"Type": 1, "Id": 41, "X": 533, "Y": 149, "W": 20, "H": 20},
        {"Type": 1, "Id": 42, "X": 559, "Y": 149, "W": 20, "H": 20},
        {"Type": 1, "Id": 43, "X": 585, "Y": 149, "W": 20, "H": 20},
        {"Type": 1, "Id": 44, "X": 507, "Y": 174, "W": 20, "H": 20},
        {"Type": 1, "Id": 45, "X": 533, "Y": 174, "W": 20, "H": 20},
        {"Type": 1, "Id": 46, "X": 559, "Y": 174, "W": 20, "H": 20},
        {"Type": 1, "Id": 47, "X": 585, "Y": 174, "W": 20, "H": 20}
    ]
},
{
    "Name": "ScreenBank",
    "Controls": [
        {"Type": 4, "Id": 0, "Lin": 4, "Char": 20, "X": 296, "Y": 120, "W": 140, "H": 50}
    ],

```

## DODATNI MATERIJALI

---

```
{
    "Type": 1, "Id": 48, "X": 286, "Y": 58, "W": 16, "H": 16},
    {"Type": 1, "Id": 49, "X": 334, "Y": 58, "W": 16, "H": 16},
    {"Type": 1, "Id": 50, "X": 382, "Y": 58, "W": 16, "H": 16},
    {"Type": 1, "Id": 51, "X": 430, "Y": 58, "W": 16, "H": 16},
    {"Type": 3, "Id": 0, "X": 279, "Y": 200, "W": 32, "H": 35},
    {"Type": 3, "Id": 1, "X": 327, "Y": 200, "W": 32, "H": 35},
    {"Type": 3, "Id": 2, "X": 375, "Y": 200, "W": 32, "H": 35},
    {"Type": 3, "Id": 3, "X": 423, "Y": 200, "W": 32, "H": 35}
]
},
{
    "Name": "JogBank",
    "Controls":
    [
        {"Type": 3, "Id": 4, "X": 671, "Y": 75, "W": 130, "H": 130}
    ]
},
{
    "Name": "JoyBank",
    "Controls":
    [
        {"Type": 5, "Id": 1, "IdA1": 16, "IdA2": 17, "IdS": 52, "X": 729, "Y": 321, "W": 45, "H": 45}
    ]
},
],
"PersistentVariables":
[
    {"Name": "THRESHOLD", "Id": 1},
    {"Name": "DEBOUNCE", "Id": 2},
    {"Name": "ENCMS", "Id": 4}
],
"Addons":
[
    {"Name": "FOOTWERK", "AddonAddress": 0}
]
}
```

Primer koda 2: Primer korisničkog profila

```
{
    "ConnectedDevices":
    {
        "53": {
            "COMPort": null,
            "DeviceId": 53,
            "Controls":
            {
                "Analog":
                {
                    "0": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {""

```

## DODATNI MATERIJALI

---

```
        "PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [1, 0]}]},  
    "1": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [2, 0]}},  
    "2": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [3, 0]}},  
    "3": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [4, 0]}},  
    "4": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [9, 0]}},  
    "5": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [10, 0]}},  
    "6": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [11, 0]}},  
    "7": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [12, 0]}},  
    "8": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [5, 0]}},  
    "9": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [6, 0]}},  
    "10": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [7, 0]}},  
    "11": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [8, 0]}},  
    "12": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [13, 0]}},  
    "13": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [14, 0]}},  
    "14": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [15, 0]}},  
    "15": {"BankName": "AnalogBank", "Mappings": {"AnalogBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [16, 0]}},  
    "16": {"BankName": "JoyBank", "Mappings": {}},  
    "17": {"BankName": "JoyBank", "Mappings": {}},  
    "18": {"BankName": "AnalogPedal", "Mappings": {"AnalogPedal1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCAnalog", "args": [17, 0]}}}
```

```
        "args": [100, 0]}]}
},
"Button": {
  "0": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCMomentaryButton", "args": [32, 0]}},,
  "1": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCMomentaryButton", "args": [33, 0]}},,
  "2": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCMomentaryButton", "args": [34, 0]}},,
  "3": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCMomentaryButton", "args": [35, 0]}},,
  "4": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCToggleButton", "args": [36, 0]}},,
  "5": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCToggleButton", "args": [37, 0]}},,
  "6": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCToggleButton", "args": [38, 0]}},,
  "7": {"BankName": "ButtonBank", "Mappings": {"ButtonBank1": {"PluginName": "MidiPlugin", "FunctionName": "MidiCCToggleButton", "args": [39, 0]}},,
  "8": {"BankName": "ButtonBank", "Mappings": {}},
  "9": {"BankName": "ButtonBank", "Mappings": {}},
  "10": {"BankName": "ButtonBank", "Mappings": {}},
  "11": {"BankName": "ButtonBank", "Mappings": {}},
  "12": {"BankName": "ButtonBank", "Mappings": {}},
  "13": {"BankName": "ButtonBank", "Mappings": {}},
  "14": {"BankName": "ButtonBank", "Mappings": {}},
  "15": {"BankName": "ButtonBank", "Mappings": {}},
  "16": {"BankName": "ButtonBank", "Mappings": {}},
  "17": {"BankName": "ButtonBank", "Mappings": {}},
  "18": {"BankName": "ButtonBank", "Mappings": {}},
  "19": {"BankName": "ButtonBank", "Mappings": {}},
  "20": {"BankName": "ButtonBank", "Mappings": {}},
  "21": {"BankName": "ButtonBank", "Mappings": {}},
  "22": {"BankName": "ButtonBank", "Mappings": {}},
  "23": {"BankName": "ButtonBank", "Mappings": {}},
  "24": {"BankName": "ButtonBank", "Mappings": {}},
  "25": {"BankName": "ButtonBank", "Mappings": {}},
  "26": {"BankName": "ButtonBank", "Mappings": {}},
  "27": {"BankName": "ButtonBank", "Mappings": {}},
  "28": {"BankName": "ButtonBank", "Mappings": {}},
  "29": {"BankName": "ButtonBank", "Mappings": {}},
  "30": {"BankName": "ButtonBank", "Mappings": {}},
  "31": {"BankName": "ButtonBank", "Mappings": {}},
```

```

    "32": {"BankName": "ButtonBank", "Mappings": {}},
    "33": {"BankName": "ButtonBank", "Mappings": {}},
    "34": {"BankName": "ButtonBank", "Mappings": {}},
    "35": {"BankName": "ButtonBank", "Mappings": {}},
    "36": {"BankName": "ButtonBank", "Mappings": {}},
    "37": {"BankName": "ButtonBank", "Mappings": {}},
    "38": {"BankName": "ButtonBank", "Mappings": {}},
    "39": {"BankName": "ButtonBank", "Mappings": {}},
    "40": {"BankName": "ButtonBank", "Mappings": {}},
    "41": {"BankName": "ButtonBank", "Mappings": {}},
    "42": {"BankName": "ButtonBank", "Mappings": {}},
    "43": {"BankName": "ButtonBank", "Mappings": {}},
    "44": {"BankName": "ButtonBank", "Mappings": {}},
    "45": {"BankName": "ButtonBank", "Mappings": {}},
    "46": {"BankName": "ButtonBank", "Mappings": {}},
    "47": {"BankName": "ButtonBank", "Mappings": {}},
    "48": {"BankName": "ScreenBank", "Mappings": {}},
    "49": {"BankName": "ScreenBank", "Mappings": {}},
    "50": {"BankName": "ScreenBank", "Mappings": {}},
    "51": {"BankName": "ScreenBank", "Mappings": {}},
    "52": {"BankName": "JoyBank", "Mappings": {}},
    "53": {"BankName": "DigitalPedals", "Mappings": {"DigitalPedals1": {
        "PluginName": "MidiPlugin", "FunctionName": "MidiCCToggleButton", "args": [101, 0]}}},
    "54": {"BankName": "DigitalPedals", "Mappings": {"DigitalPedals1": {
        "PluginName": "MidiPlugin", "FunctionName": "MidiCCMomentaryButton", "args": [102, 0]}}}
},
"RGBLED": {
    "0": {"BankName": "ButtonBank", "Mappings": {}},
    "1": {"BankName": "ButtonBank", "Mappings": {}},
    "2": {"BankName": "ButtonBank", "Mappings": {}},
    "3": {"BankName": "ButtonBank", "Mappings": {}},
    "4": {"BankName": "ButtonBank", "Mappings": {}},
    "5": {"BankName": "ButtonBank", "Mappings": {}},
    "6": {"BankName": "ButtonBank", "Mappings": {}},
    "7": {"BankName": "ButtonBank", "Mappings": {}},
    "8": {"BankName": "ButtonBank", "Mappings": {}},
    "9": {"BankName": "ButtonBank", "Mappings": {}},
    "10": {"BankName": "ButtonBank", "Mappings": {}},
    "11": {"BankName": "ButtonBank", "Mappings": {}},
    "12": {"BankName": "ButtonBank", "Mappings": {}},
    "13": {"BankName": "ButtonBank", "Mappings": {}},
    "14": {"BankName": "ButtonBank", "Mappings": {}},
    "15": {"BankName": "ButtonBank", "Mappings": {}}
},
"LCD": {
    "0": {"BankName": "ScreenBank", "Mappings": {}}
},

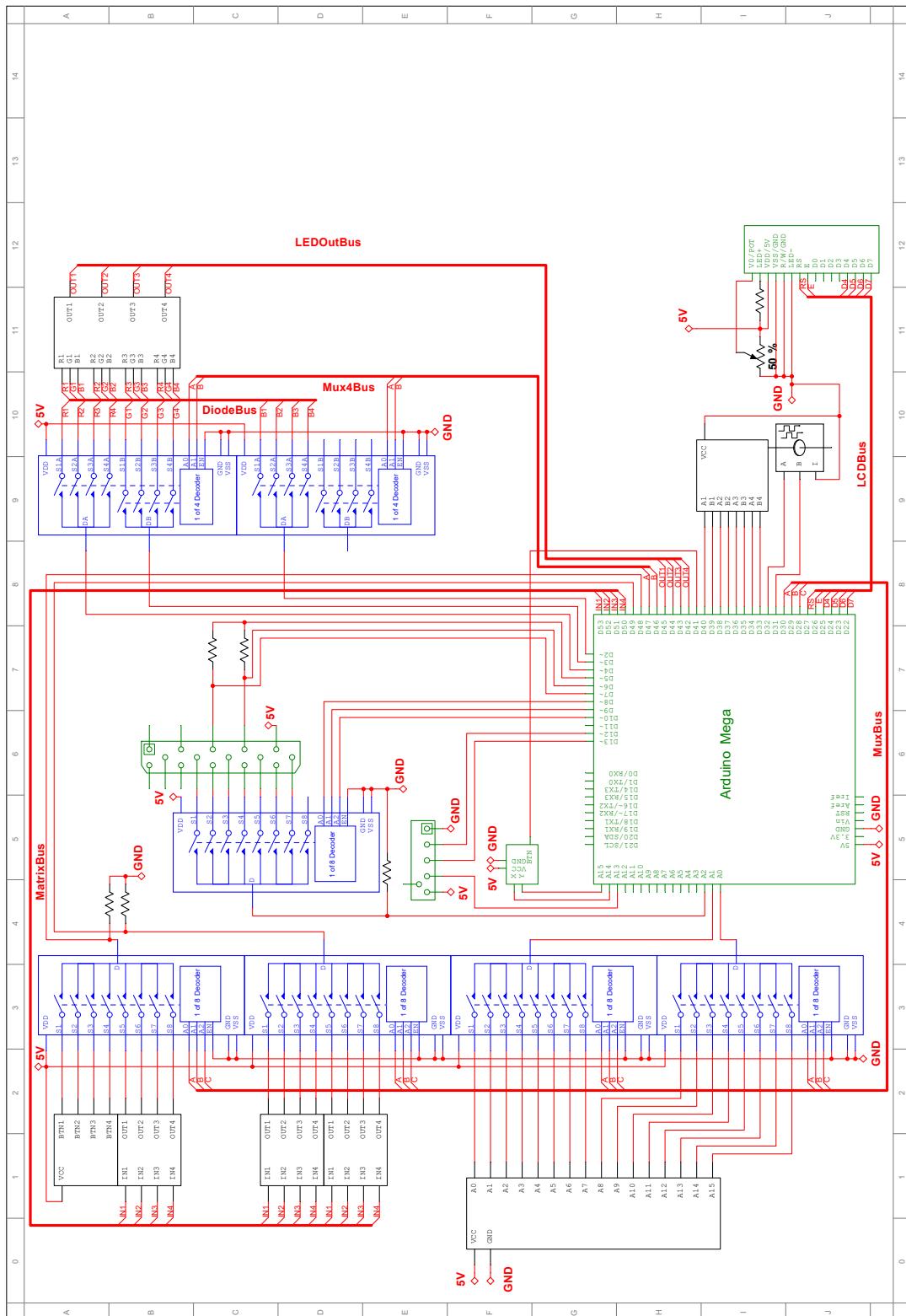
```

## DODATNI MATERIJALI

---

```
"Encoder":  
{  
    "0": {"BankName": "ScreenBank", "Mappings": {}},  
    "1": {"BankName": "ScreenBank", "Mappings": {}},  
    "2": {"BankName": "ScreenBank", "Mappings": {}},  
    "3": {"BankName": "ScreenBank", "Mappings": {}},  
    "4": {"BankName": "JogBank", "Mappings": {}}  
}  
},  
"ActivePage":  
{  
    "AnalogBank": "AnalogBank1",  
    "ButtonBank": "ButtonBank1",  
    "ScreenBank": "ScreenBank1",  
    "JogBank": "JogBank1",  
    "JoyBank": "JoyBank1",  
    "AnalogPedal": "AnalogPedal1",  
    "DigitalPedals": "DigitalPedals1"  
}  
}  
}  
},  
"PluginStates": {}  
}
```

## DODATNI MATERIJALI



Slika 1: Šema elektronike

# Literatura

- [1] Arduino AG. *Arduino*. 2019. URL: <https://www.arduino.cc/>.
- [2] Arduino AG. *Reading Rotary Encoders*. 2019. URL: <https://playground.arduino.cc/Main/RotaryEncoders/>.
- [3] MIDI Manufacturers Association. *MIDI*. 2019. URL: <https://www.midi.org/>.
- [4] Dave Dribin. *Keyboard Matrix Help*. 2000. URL: [https://www.dribin.org/dave/keyboard/one\\_html/](https://www.dribin.org/dave/keyboard/one_html/).
- [5] SparkFun Electronics. *Button Pad Hookup Guide*. 2019. URL: <https://learn.sparkfun.com/tutorials/button-pad-hookup-guide>.
- [6] Tobias Erichsen. *virtualMIDI SDK*. 2019. URL: <http://www.tobias-erichsen.de/software/virtualmidi/virtualmidi-sdk.html>.
- [7] Number Eight Innovation. *THREE METHODS TO FILTER NOISY ARDUINO MEASUREMENTS*. 2019. URL: <https://www.megunolink.com/articles/coding/3-methods-filter-noisy-arduino-measurements/>.
- [8] EETech Media LLC. *Switch Bounce and How to Deal with It*. 2019. URL: <https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/>.
- [9] Blackmagic Design Pty. Ltd. *Blackmagic Design*. 2019. URL: <https://www.blackmagicdesign.com/>.
- [10] COMPULITE SYSTEMS LTD. *Compulite*. 2019. URL: <http://www.compulite.com/>.
- [11] Microsoft. *Microsoft Developer Network*. 2019. URL: <https://msdn.microsoft.com>.
- [12] Microsoft. *Using threads and threading*. 2019. URL: <https://docs.microsoft.com/en-us/dotnet/standard/threading/using-threads-and-threading>.

## *LITERATURA*

---

- [13] Lazar Mitrović. *Od NAND-a do procesora*. 2018. URL: [http://www.csnedelja.mg.edu.rs/static/resources/v4.0/uto3\\_odnanda\\_lm.pdf](http://www.csnedelja.mg.edu.rs/static/resources/v4.0/uto3_odnanda_lm.pdf).
- [14] JSON org. *Introducing JSON*. 2019. URL: <https://www.json.org/>.

RAČUNARSKA GIMNAZIJA  
Lazar Premović

maturski rad iz predmeta:  
**Napredne tehnike programiranja**

# ČOVEK-MAŠINA INTERFEJS ZA PRIMENU U MUZICI

---

Professional Hardware Interfaces - Generic Professional Controller "PHI - GPC"

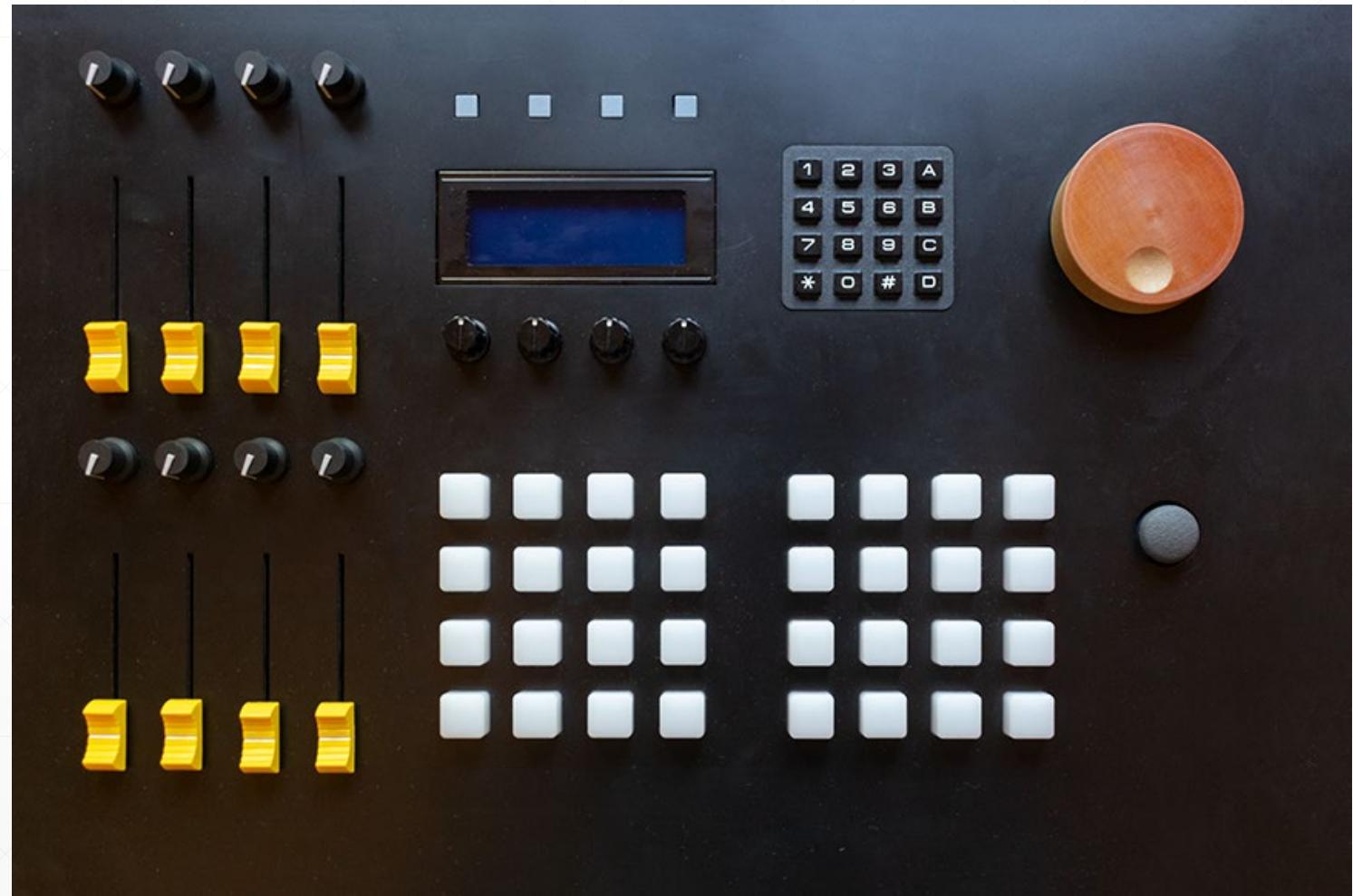
Beograd, jun 2019.

Mentor: Prof. dr Filip Marić

# GPC Kontroler

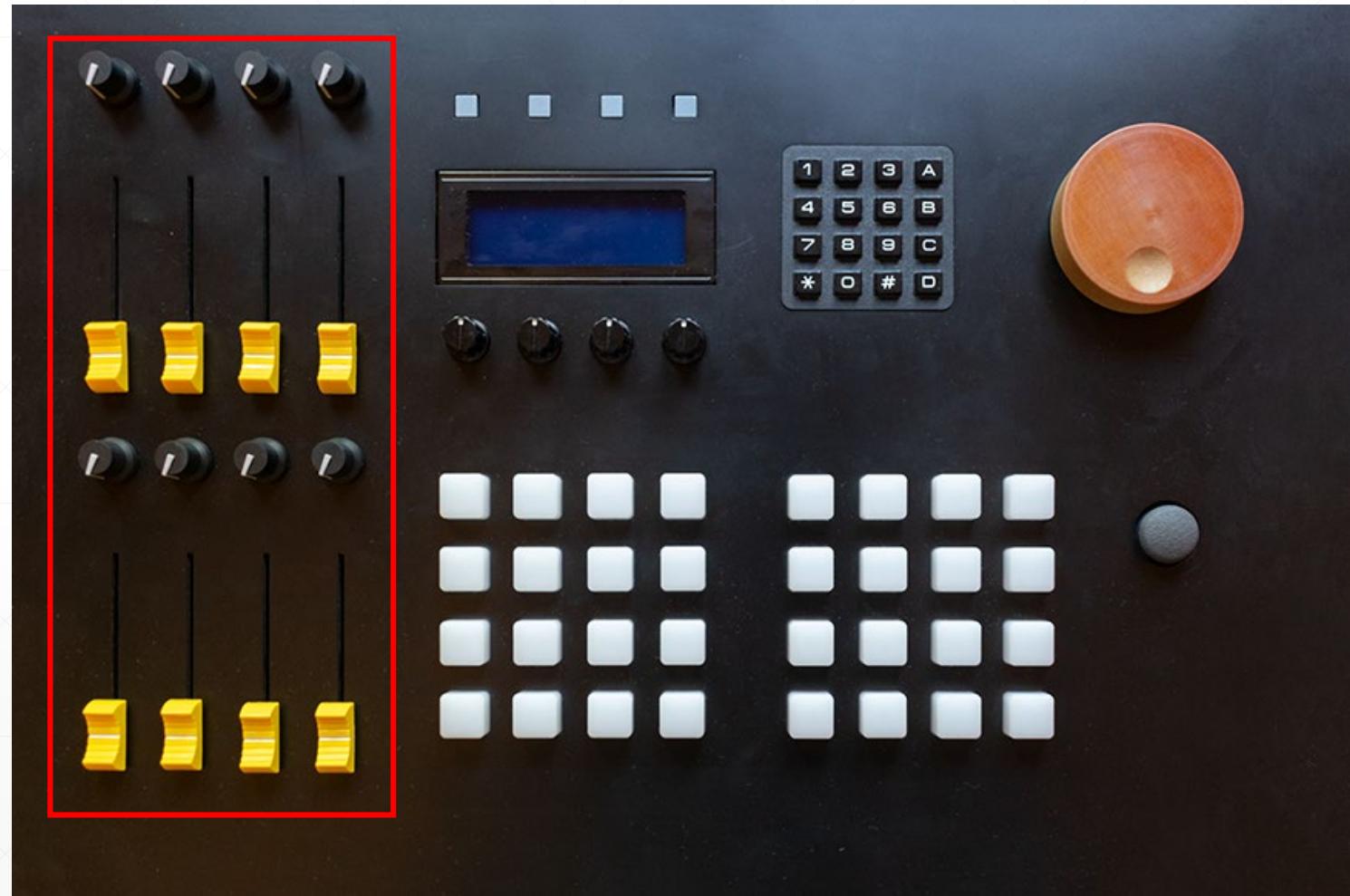


# GPC Kontroler



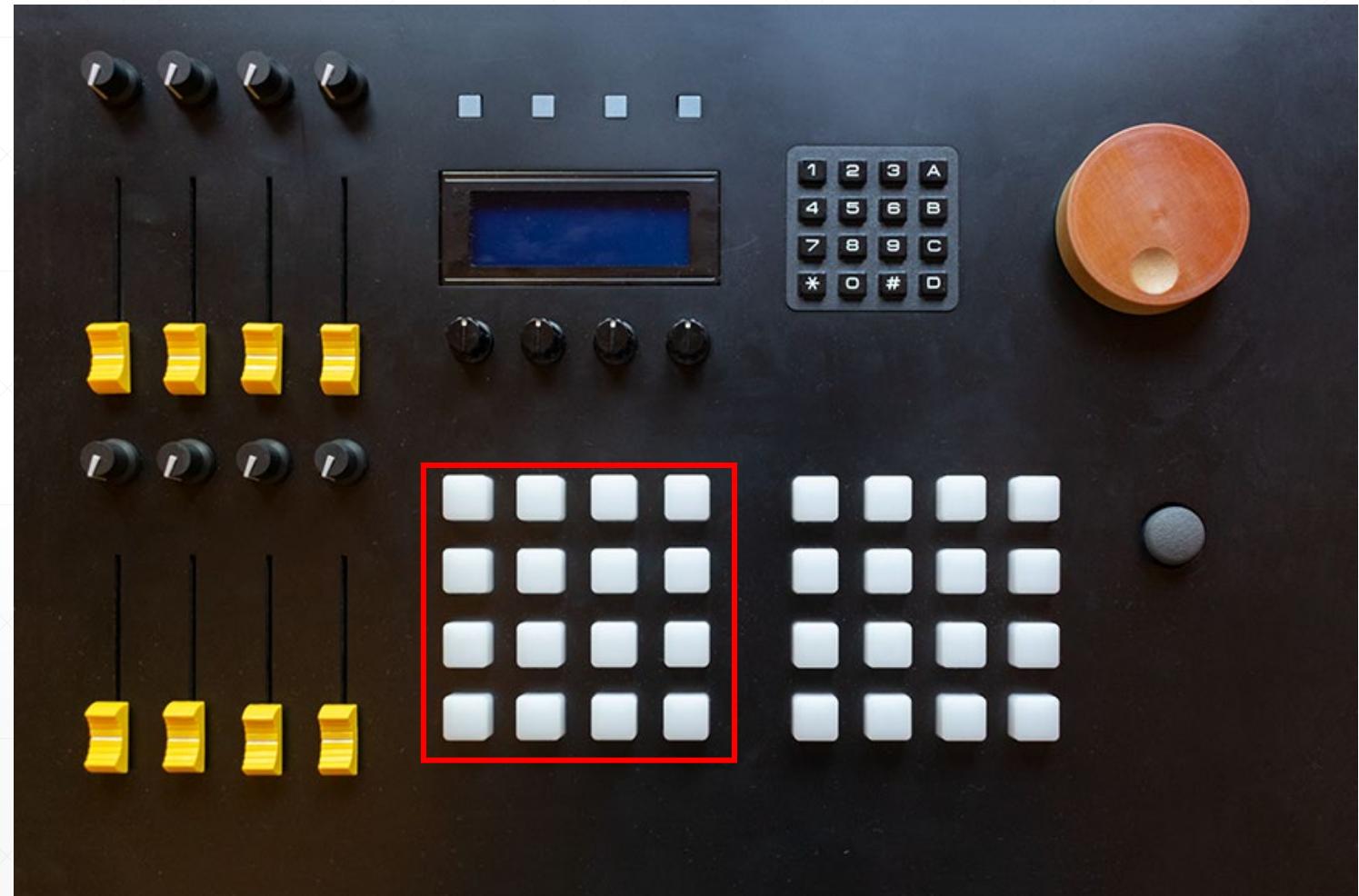
# GPC Kontroler

- 16 Analognih Kontrola



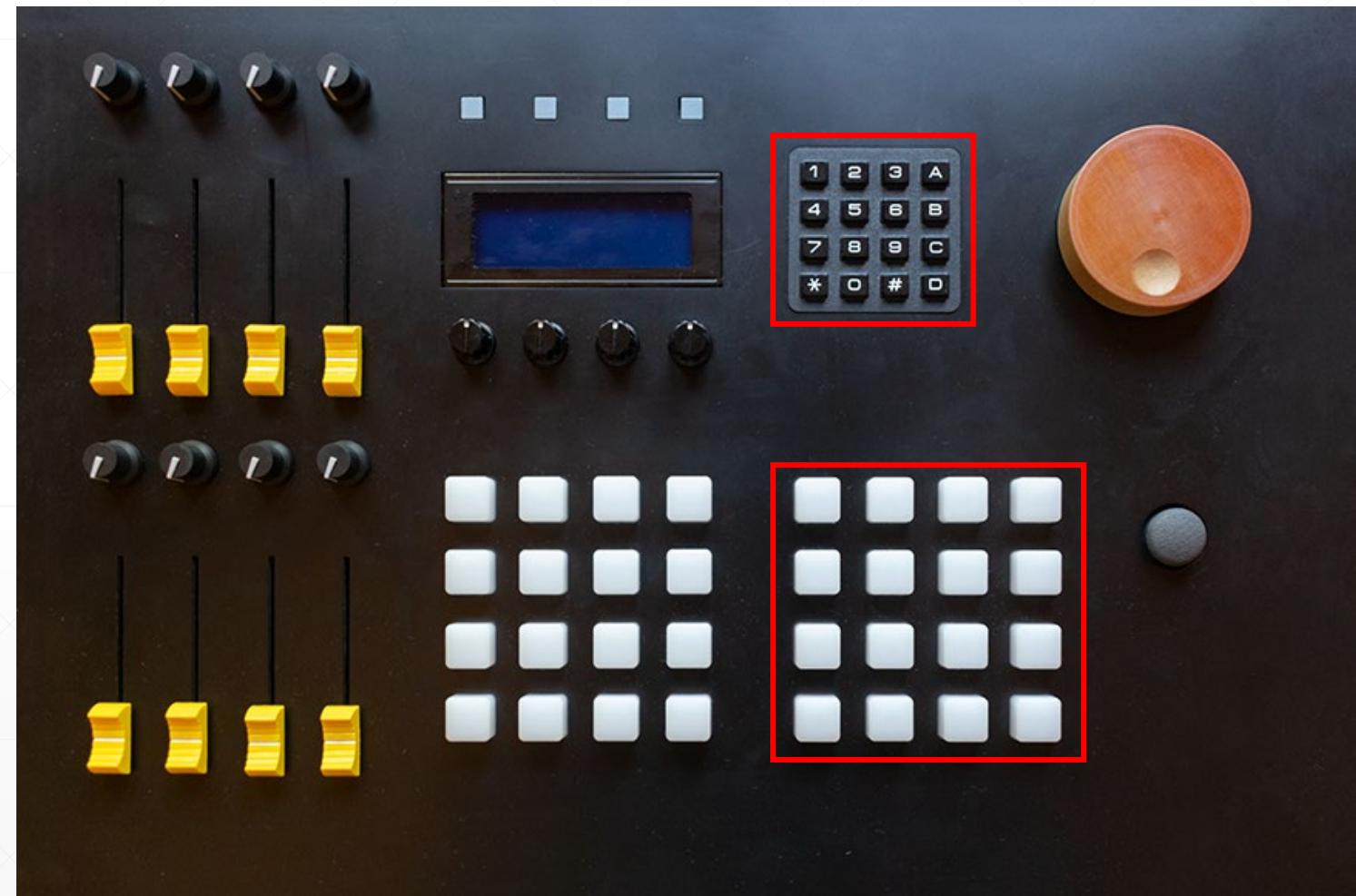
# GPC Kontroler

- 16 Analognih Kontrola
- Matrix tastera sa RGB LED diodom ispod svakog tastera



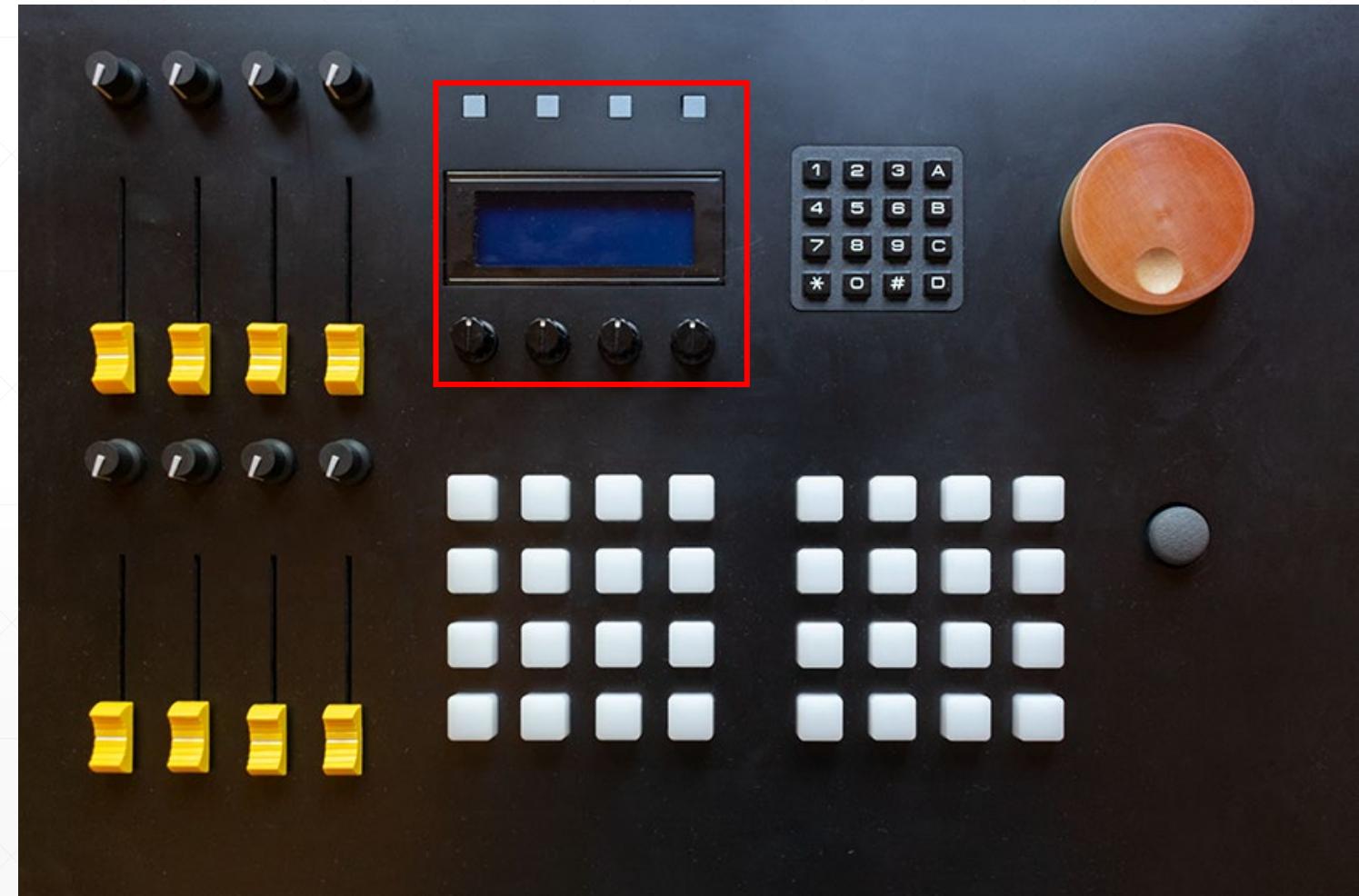
# GPC Kontroler

- 16 Analognih Kontrola
- Matrica tastera sa RGB LED diodom ispod svakog tastera
- Još dve matrice tastera



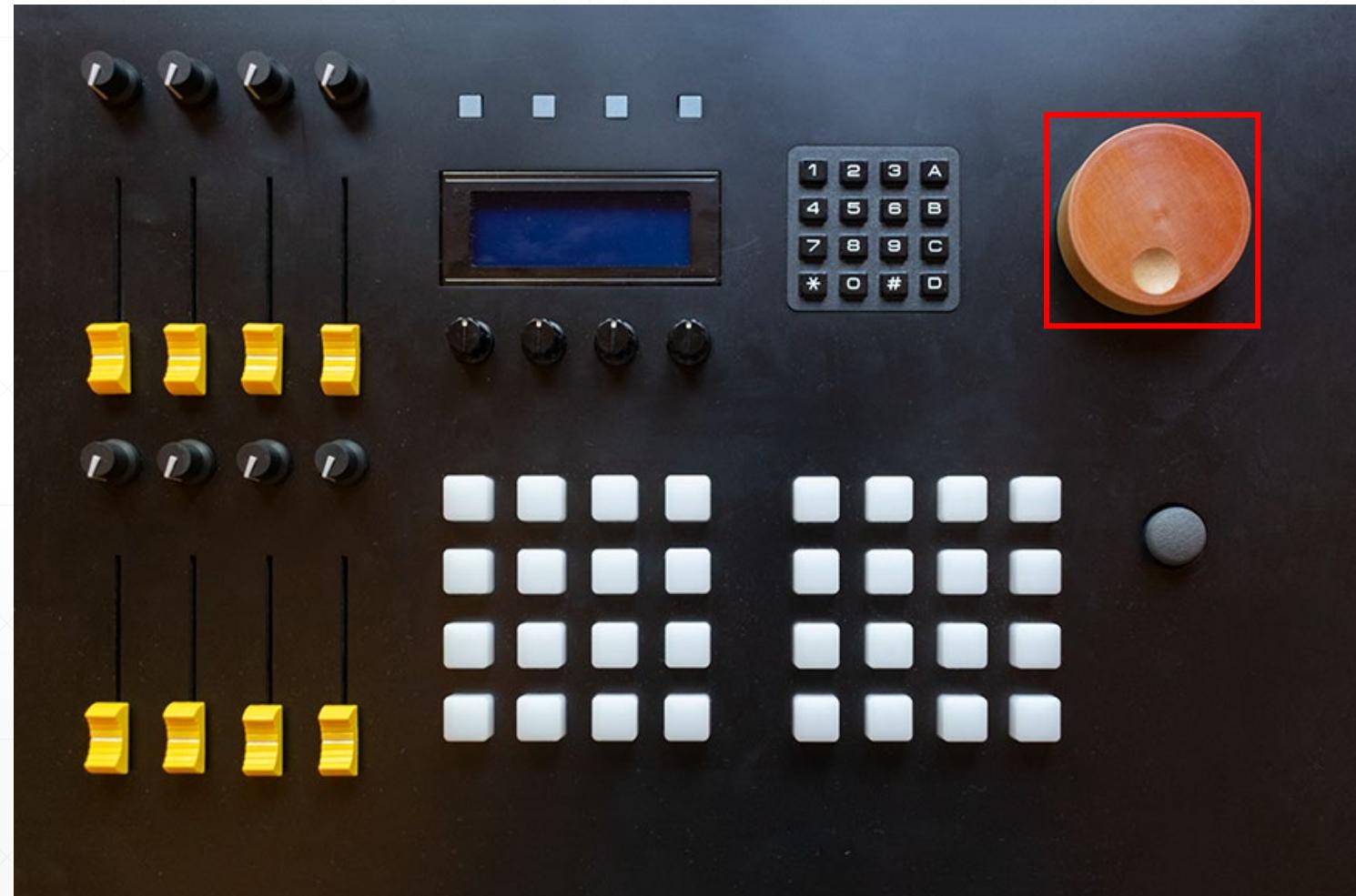
# GPC Kontroler

- 16 Analognih Kontrola
- Matrica tastera sa RGB LED diodom ispod svakog tastera
- Još dve matrice tastera
- LCD displej sa 4 tastera i 4 rotaciona enkodera



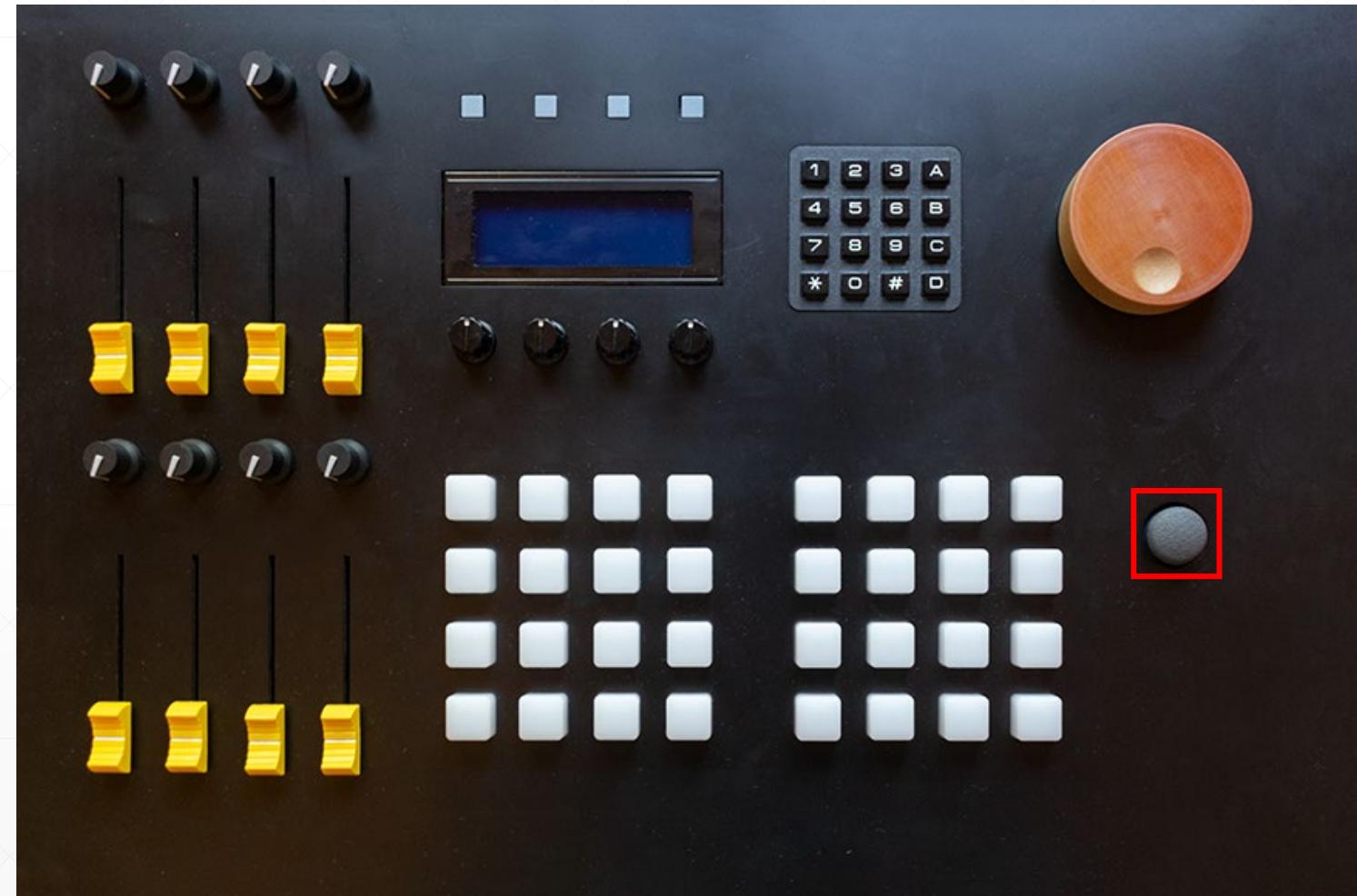
# GPC Kontroler

- 16 Analognih Kontrola
- Matrica tastera sa RGB LED diodom ispod svakog tastera
- Još dve matrice tastera
- LCD displej sa 4 tastera i 4 rotaciona enkodera
- Točak za brzu navigaciju (Jog wheel)



# GPC Kontroler

- 16 Analognih Kontrola
- Matrica tastera sa RGB LED diodom ispod svakog tastera
- Još dve matrice tastera
- LCD displej sa 4 tastera i 4 rotaciona enkodera
- Točak za brzu navigaciju (Jog wheel)
- Džoystik



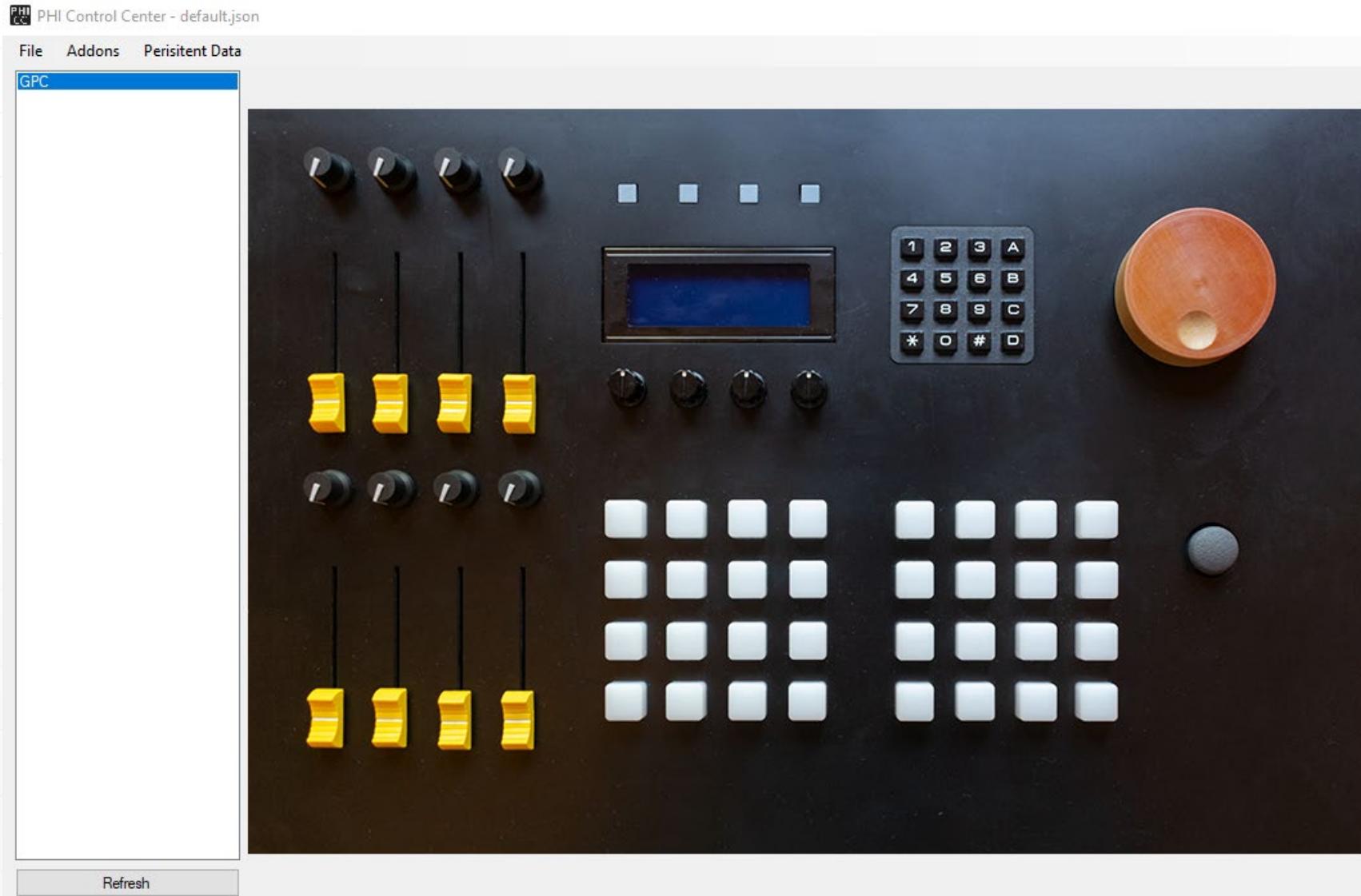
# GHP Protokol

*CommandByte, ChannelByte, [0 - 20]DataBytes*

Komandni bajt	Komanda	Kanal	Opis bajtova sa podacima
'B' 66	Taster je pritisnut	Indeks tastera	NEMA
'R' 82	Taster je otpušten	Indeks tastera	NEMA
'A' 65	Analogna kontrola je pomerena	Indeks analogne kontrole	2 bajta. Nova vrednost analogue kontrole, Little Endian enkodiranje
'T' 73	Rotacioni enkoder je inkrementovan	Indeks enkodera	NEMA
'D' 68	Rotacioni enkoder je dekrementovan	Indeks enkodera	NEMA
'L' 76	Podesi LED	Indeks diode	3 bajta. R, G, B
'T' 108	Podesi LCD	Broj linije ekrana	20 bajtova. Tekst
'P' 80	Podesi trajni podatak	Memorijska adresa	1 bajt. Vrednost
'p' 112	Zahtev za vrednost trajnog podatka	Memorijska adresa	NEMA
'r' 114	Odgovor na zahtevanu vrednost.	Memorijska adresa	1 bajt. Vrednost
'S' 83	Sistemska komanda (videti tabelu 2.2)	Vrsta sistemske komande	Zavisi od vrste sistemske komande

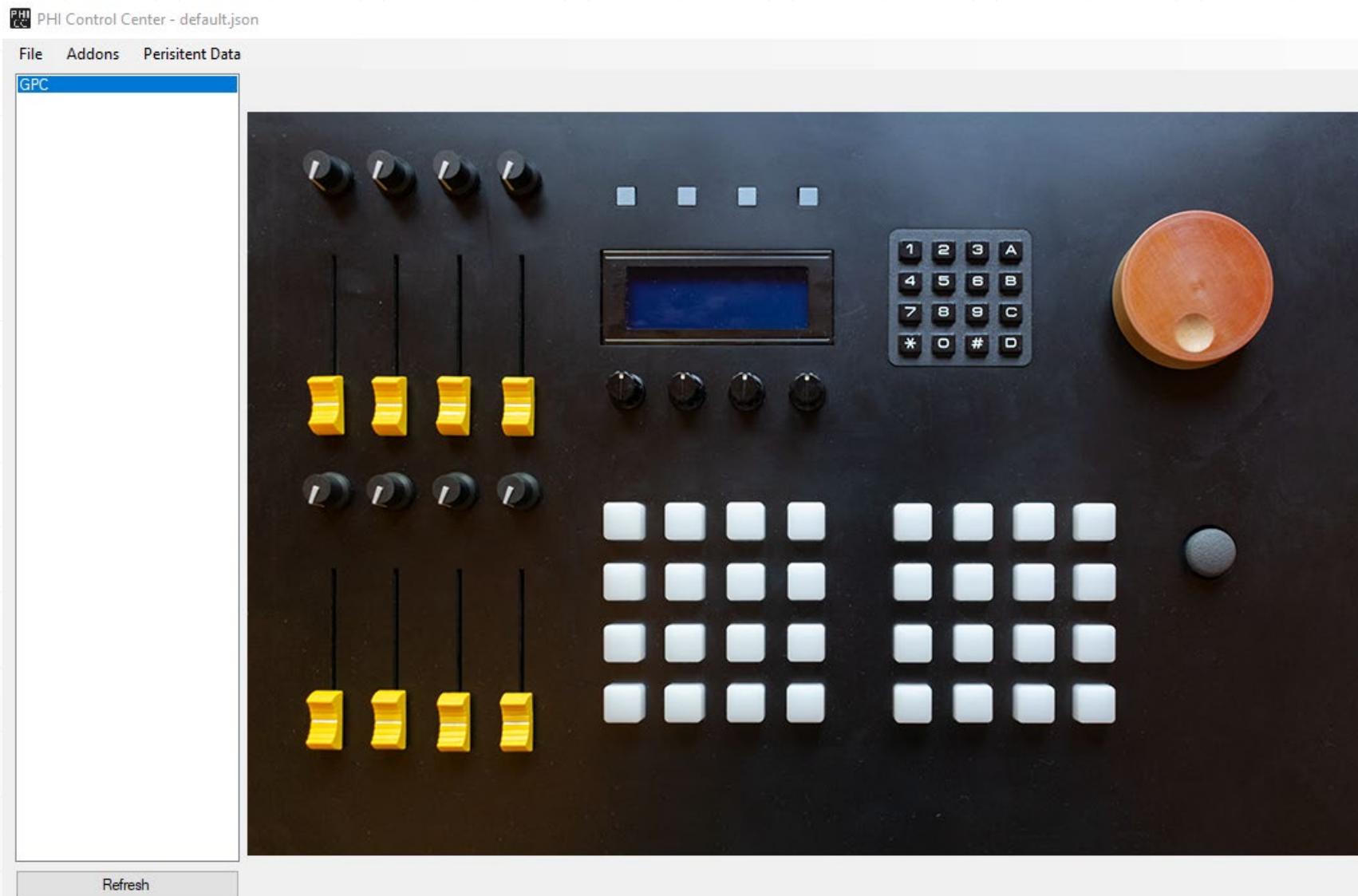
- USB serijska komunikacija
- 11520 baud-rate
- Komandni bajtovi iz ASCII alfanumeričkog opsega

# PHI Control Center – Konfiguracija



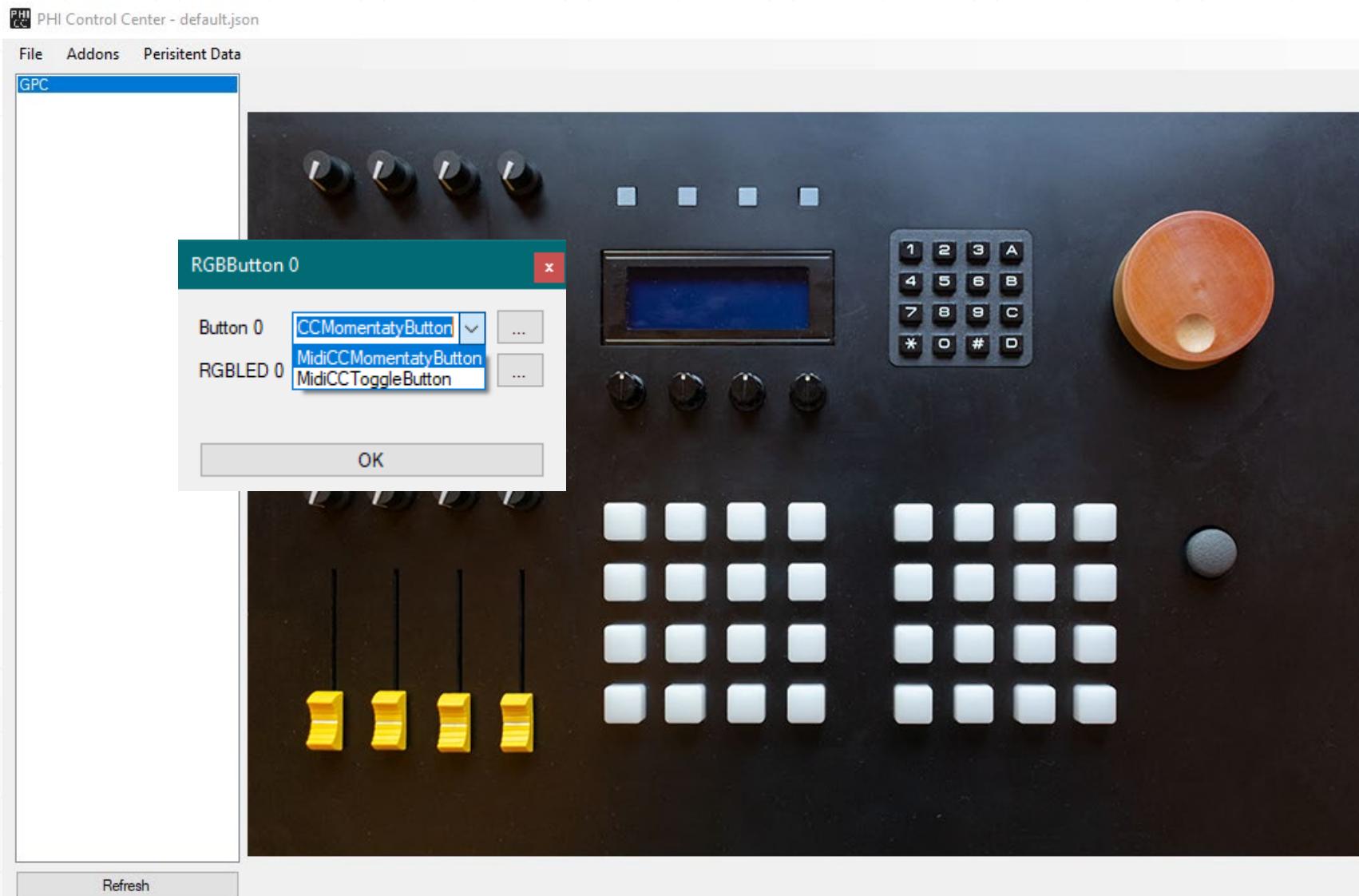
# PHI Control Center – Konfiguracija

- Definicija uređaja



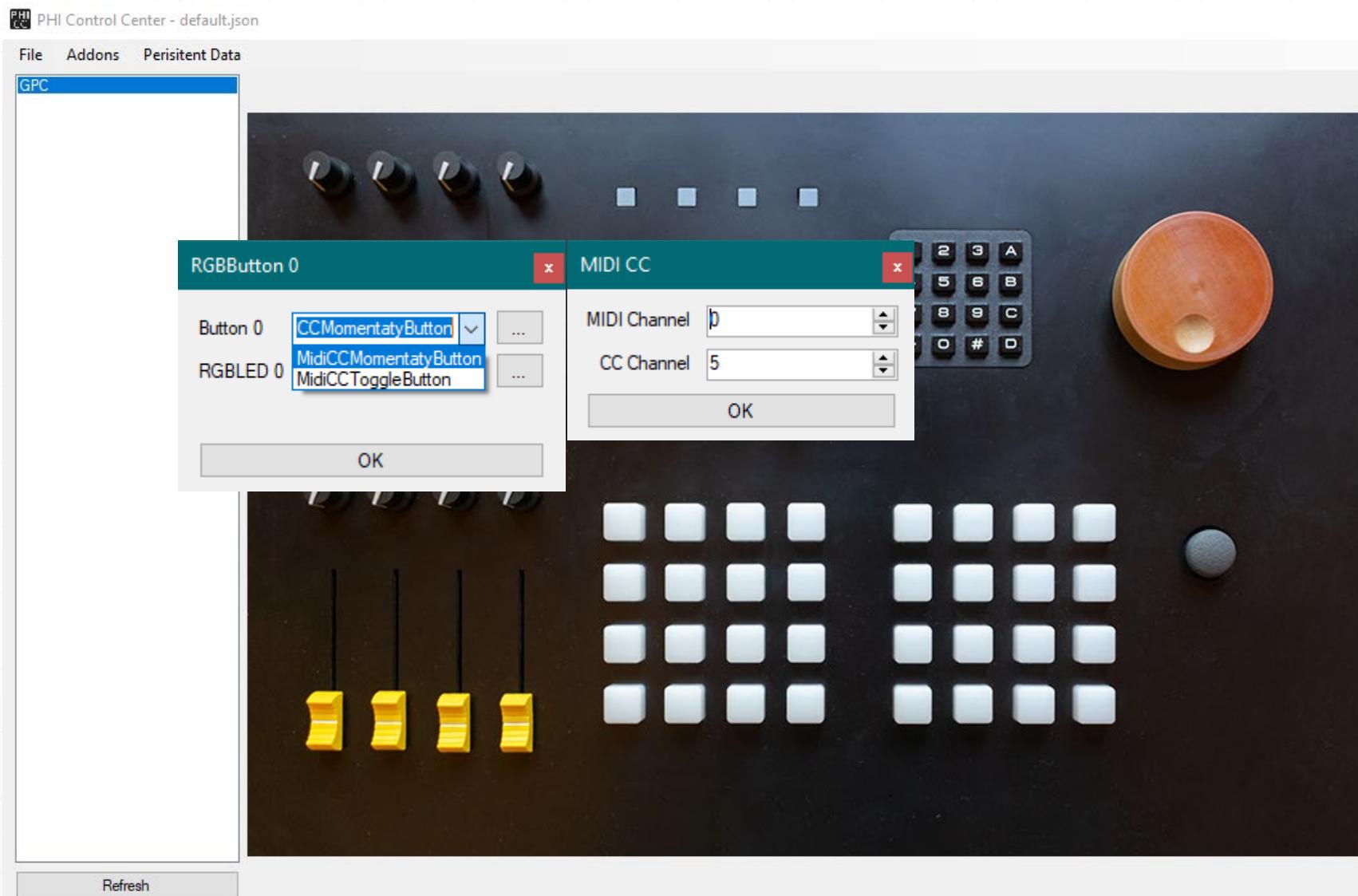
# PHI Control Center – Konfiguracija

- Definicija uređaja
- Izbor kontrole
- Izbor Funkcije



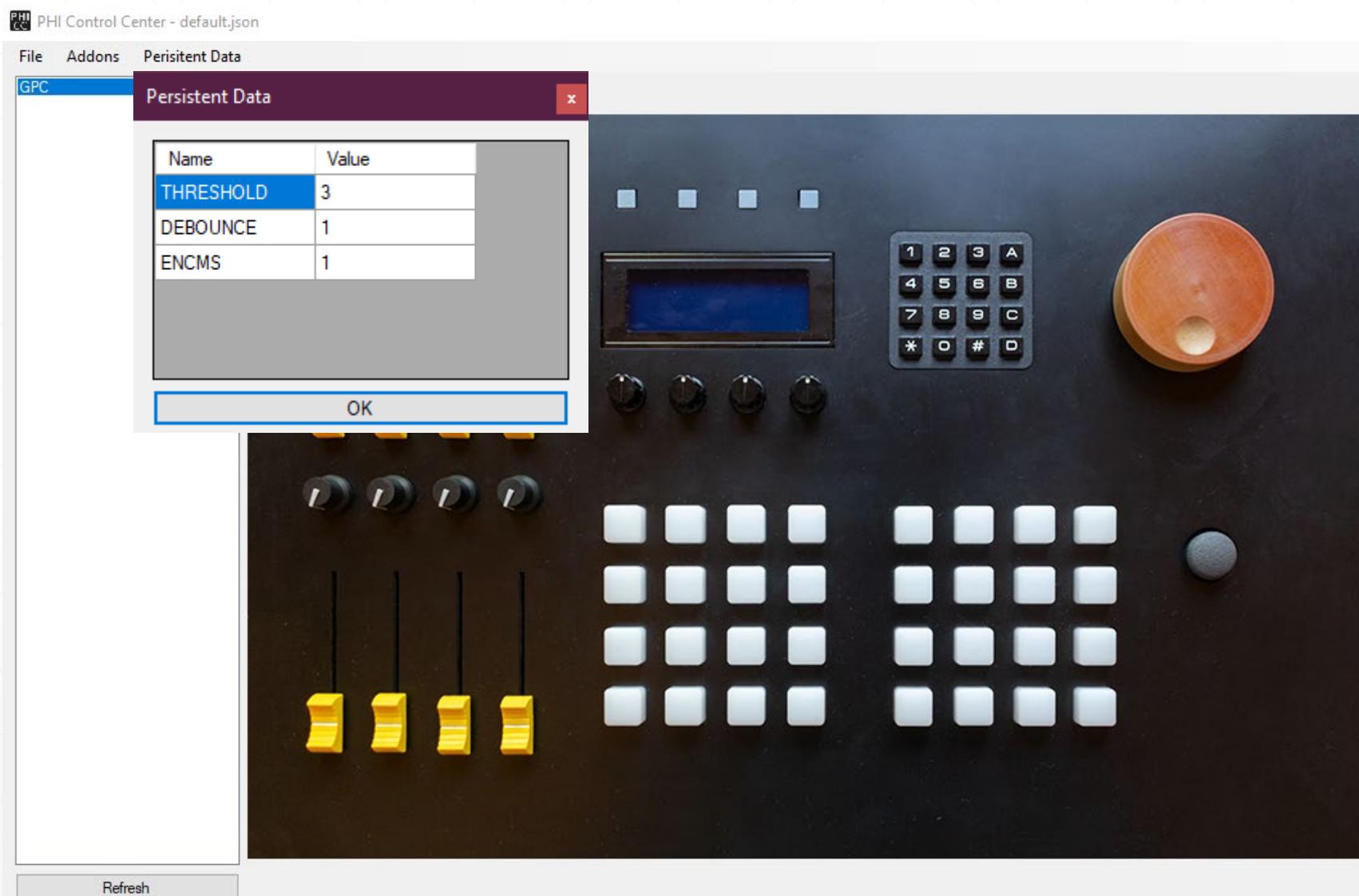
# PHI Control Center – Konfiguracija

- Definicija uređaja
- Izbor kontrole
- Izbor Funkcije
- Podešavanje parametara funkcije



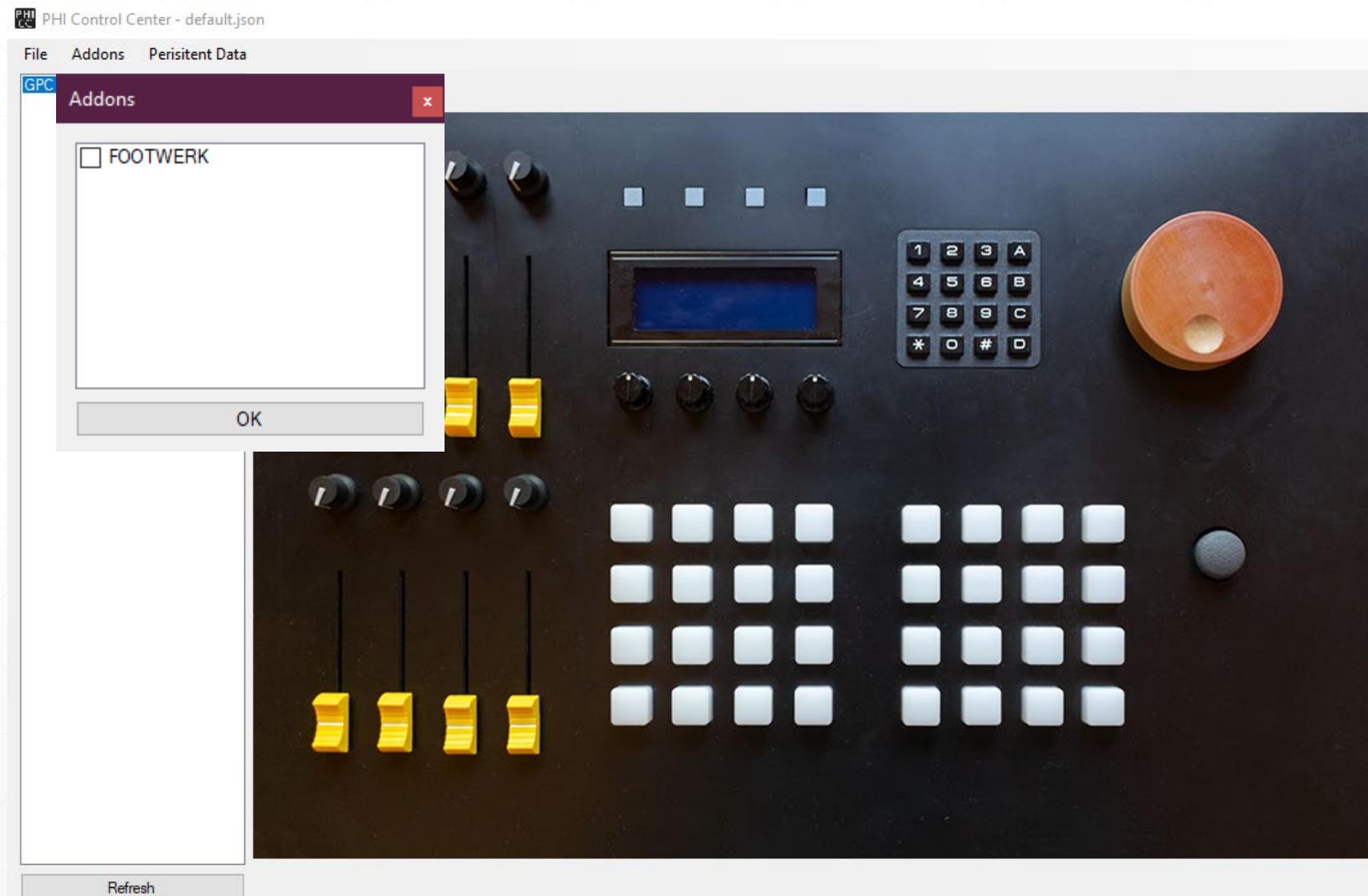
# PHI Control Center – Konfiguracija

- Definicija uređaja
- Izbor kontrole
- Izbor Funkcije
- Podešavanje parametara funkcije
- Podešavanje trajnih podataka



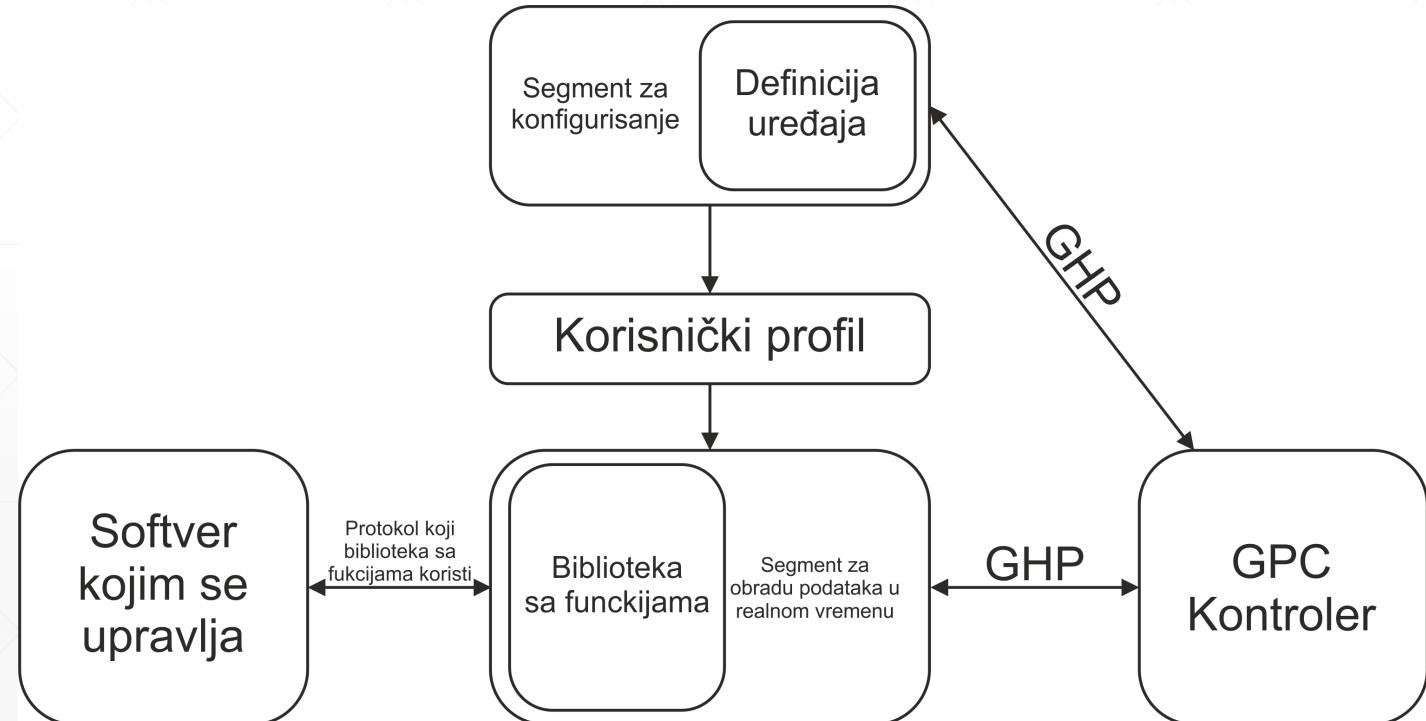
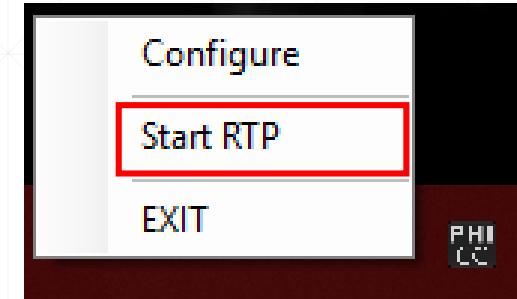
# PHI Control Center – Konfiguracija

- Definicija uređaja
- Izbor kontrole
- Izbor Funkcije
- Podešavanje parametara funkcije
- Podešavanje trajnih podataka
- Aktivacija opcionih dodataka



# PHI Control Center – Obrada u realnom vremenu

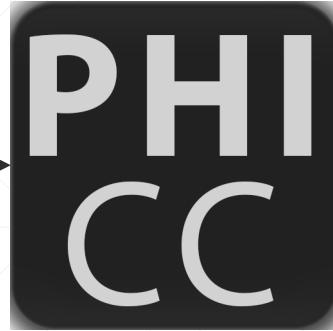
- Na korisniku je samo da startuje nit koja vrši obradu u realnom vremenu klikom na dugme Start RTP
- Korisnički profil
- Biblioteka sa funkcijama
- MidiPlugin
  - MidiCCAnalog
  - MidiCCMomentaryButton
  - MidiCCToggleButton
  - MidiCCEncoder
  - MidiCCLED
  - MidiCCLCD
  - MidiPLAY
  - MidiCONT
  - MidiSTOP
  - MidiENCSHIFT



# DEMO



MIDI



GHP



- Kontrola nivoa kanala
- Kontrola stereo pozicije kanala
- Kontrola utišavanja kanala
- Kontrola zaobilaženja efekata
- Kontrola parametara efekata
- Upravljanje transport sekcijom

I možda malo magije

# Završne reči i pitanja

---

# Hvala na pažnji

---

Lazar Premović

[lazar2premovic@gmail.com](mailto:lazar2premovic@gmail.com)