Primjena strojnog učenja na mikroupravljačkim sustavima

Katalinić, Helena

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

Sveučilišni studij

Primjena strojnog učenja na mikroupravljačkim sustavima

Završni rad

Helena Katalinić

Osijek, 2024.

SADRŽAJ

1. UVOD

U modernom svijetu okruženi smo mnoštvom podataka koji su relevantni u raznim područjima našeg svakidašnjeg života. Takve podatke možemo analizirati pomoću raznih znanstveno-tehnoloških alata, a jedan od najmoćnijih među njima je strojno učenje. Strojno učenje omogućuje nam da izvučemo vrijedne informacije, prepoznamo obrasce i trendove te donesemo informirane odluke, koristeći algoritme koji se sami prilagođavaju na temelju podataka. Ova grana umjetne inteligencije igra ključnu ulogu u raznim područjima kao što su zdravstvo, poslovanje, znanost, tehnologija, financije i mnogim drugima. Iskorištavanje strojnog učenja na pametan način može rezultirati boljim razumijevanjem svijeta oko nas i poboljšanjem različitih aspekata našeg života. Sve podatke koje bilježe uređaji možemo svrstavati koristeći strojno učenje.

Strojno učenje je grana umjetne inteligencije čiji je zadatak korištenje određenih algoritama koji predstavljaju imitaciju modela ljudskog učenja. Ima razne primjene, neke od kojih su istraživanja, stvaranju umjetničkih kreacija, poboljšavanju svakodnevice.

Napretkom tehnologije dolazi do stvaranja sve manjih sustava koji se stavljaju unutar ugrađenih sustava. Jedan primjer takve komponente je mikroupravljač koji je rasprostranjen u različitim aplikacijskim uređajima. S obzirom na to da su malih dimenzija, prilagodljivi i imaju nisku potrošnju energije idealan su kandidat za implementaciju modela strojnog učenja u aplikacijama gdje je potrebna lokalna obrada podataka.

U ovom radu će se predstaviti korištenje strojnog učenja primjenom posebno namijenjenog mikroupravljačkog sustava uz platformu koja ima sve potrebne funkcionalnosti implementacije zadatka. Prvi korak je izabrati adekvatan model koji je prikladan za korištenje i istrenirani model pohraniti na mikroupravljač, nakon čega dolazi testiranje samog modela.

Proces implementacije na mikroupravljač prikazati će se na modelu prepoznavanja ključnih riječi „crvena“ ili „plava“, te ako se one izgovore na razvojnoj pločici će se aktivirati odgovarajuća boja.

1.1. Zadatak završnog rada

Zadatak završnog rada je implementacija metoda strojnog učenja na mikroupravljačkim sustavima niske snage.

2. PREGLED PODRUČJA TEME

Strojno učenje postaje dio naše svakodnevnice svojim primjenama u raznim sektorima neki od kojih su poljoprivreda, financije, vojska , medicina kao i mnoge druge primjene. Stvaranjem novih rješenja za nove tehnološke izazove dolazimo do implementiranja posebne vrste modela strojnog učenja koji su karakteristični po tome što zauzimaju puno manje memorije. Razlog za stvaranje novih platformi na kojima je moguće stvarati programe koji troše malo resursa i zauzimaju malo prostora je činjenica da takvi sustavi mogu raditi iako nisu povezani s internetom.

Kada se model istrenira u okruženju kao što je na primjer Tensorflow mogu se koristiti konverteri koji će model prilagoditi mikroupravljaču i pretvoriti ga u verziju TensorFlow Lite koja je prilagođena za implementaciju na mikrokontoler i na taj način se trenirani model postavlja na mikroupravljač, te on prema datom modelu radi svoju zadaću na podatcima.

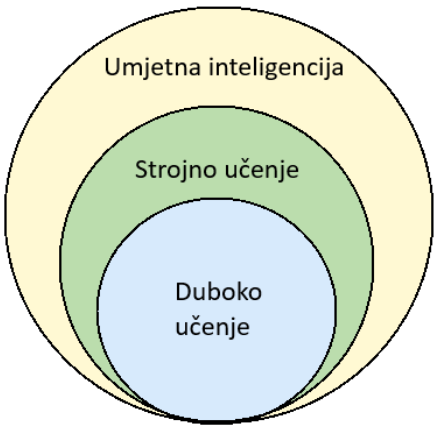
Platforme koje se mogu koristiti za implementaciju modela strojnog učenja na mikroupravljače uključuju TinyML, EML, MicroPython, Zephyr i Edge Impulse. TinyML se odnosi na primjenu strojnog učenja na uređajima s vrlo ograničenim resursima, omogućujući izvođenje ML modela uz minimalnu potrošnju energije, što je idealno za aplikacije poput prepoznavanja govora i detekcije anomalija. EML se fokusira na ugrađene sustave i omogućava implementaciju ML modela u industrijskim, medicinskim i potrošačkim uređajima. MicroPython koristi implementaciju Python 3 jezika optimiziranog za mikroupravljače, što olakšava brzo prototipiranje i razvoj IoT aplikacija. Zephyr je otvoreni RTOS za ugrađene sustave, pružajući robusnu platformu za industrijske, medicinske, potrošačke i IoT aplikacije. Edge Impulse je platforma za razvoj, učenje i implementaciju ML modela na uređajima, omogućujući treniranje modela i njihovu integraciju u ugrađene sustave za praćenje okoliša, zdravstvo, poljoprivredu i industriju. Korištenjem ovih platformi moguće je razviti sofisticirane aplikacije strojnog učenja koje rade na uređajima s ograničenim resursima bez potrebe za stalnim povezivanjem na internet.

3. TEORIJSKA PODLOGA

Ovo poglavlje daje podlogu koja je potrebna za adekvatno razumijevanje teme završnog rada i samog načina na koji se zadatak rješava.

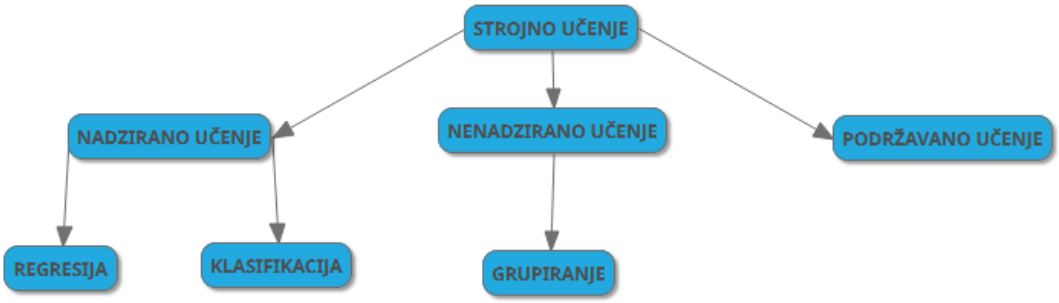
3.1. Strojno učenje

Model prima podatke iz svoje okoline i na temelju njih dolazi do određenih zaključaka kao što je na primjer klasifikacija skupa podataka u svrhu nekog istraživanja [1].



Sl. 3.1.1. Prikaz ustrojstva umjetne inteligencije [1]

Postoje različite metode strojnog učenja koje se koriste za rješavanje različitih problema analize podataka. Postoje tri metode strojnog učenja ilustrirane na dijagramu *Sl. 3.1.2.*, a to su nadzirano učenje, nenadzirano učenje i podržano učenje [2]. Nenadzirano učenje ne sadrži točne izlaze, već je cilj klasificirati elemente u određenje skupine. Podržano učenje se koristi u sustavima koji rade u okruženju i trebaju iskoristiti trenutne podatke.



Sl. 3.1.2. Prikaz vrsta strojnog učenja

Nadzirano strojno učenje dijeli se na klasifikaciju i regresiju . Klasifikacija predstavlja algoritam pogodan za sortiranje novog ulaznog podatka *Sl. 3.1.1.* Prikladna za prepoznavanje govora, prepoznavanje na temelju podataka koji se dobivaju kamerama, itd.

Regresijski algoritam se koristi kada je cilj predvidjeti numeričku vrijednost izlazne varijable na temelju ulaznih podataka. Ova tehnika se često koristi u predviđanju kontinuiranih varijabli poput cijena nekretnina, temperature ili financijskih pokazatelja.

Današnja tehnologija teži razvijanju što manjih softvera i uređaja koji s vrlo malo energije mogu obrađivati podatke u stvarnom vremenu i na taj način dolazi do razvijanja takozvanog rubnog računarstva (*engl. Edge computing*). Rubno računarstvo teži praktičnijem iskorištavanju memorije i optimalizaciji softvera gdje je projekt istreniran na oblaku *(engl. Cloud*), a samu zadaću kojoj je namijenjen izvršava direktno na uređaju kako bi se zaštitila privatnost korisnika, te što efikasnije izvršila zadaća.

3.2. Analiza mikroupravljačkog sustava

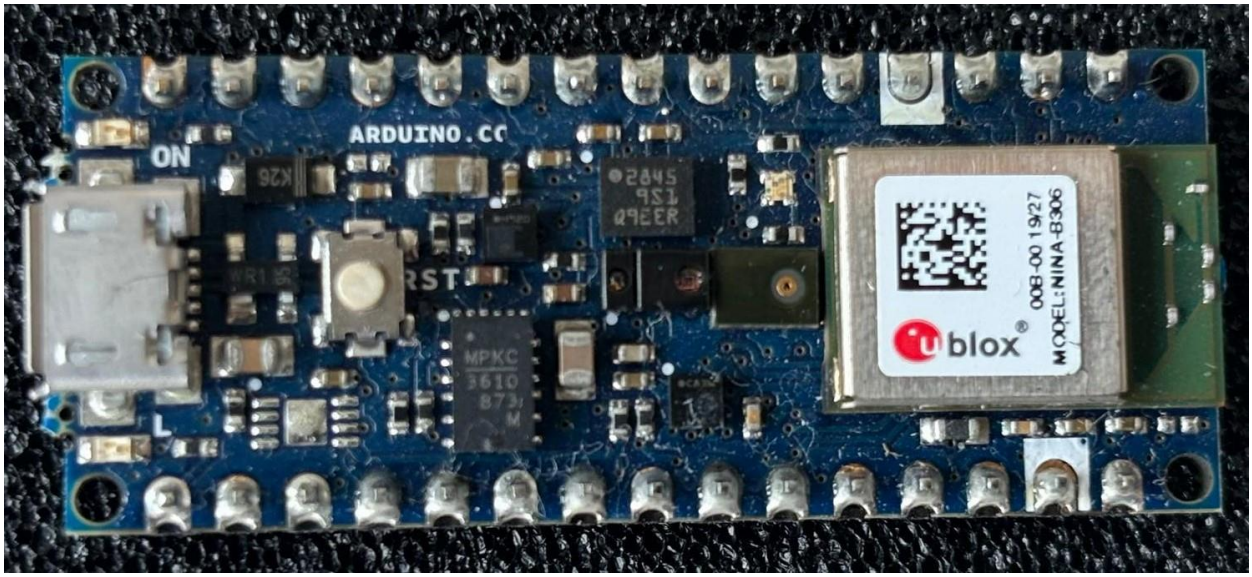
Mikroupravljači su jeftini samostalni računalni sustavi koji se pojavljuju u svakodnevnim uređajima oko nas kao što su mobiteli, mikrovalne, mašine za pranje suđa i mnoge druge [3].

U realizaciji ovog projekta koristit će se Arduino Nano 33 BLE prikazana na *Sl. 3.2.1* pločica koja sadrži različite mogućnosti kao što je detekcija gibanja objekta, bluetooth, detekcija zvuka, temperature i vlage u zraku. Najvažnije opcija je mogućnost pokretanja takozvanih „Edge Computing“ aplikacija.

Sadrži nRF52840 mikroupravljač, 32 bitni ARM Cortex-M4 CPU koji radi na 64MHz koji je pogodan za izvođenje složenijih algoritama [4].

Tehničke specifikacije :

- Mikrokontroler: Nordic nRF52840
- Radna frekvencija: 64 MHz
- Radni napon: 3.3V
- Ulazni napon: 21V
- CPU Flash memorija: 1MB
- SRAM: 256KB
- Bluetooth Low Energy (BLE) 5.0



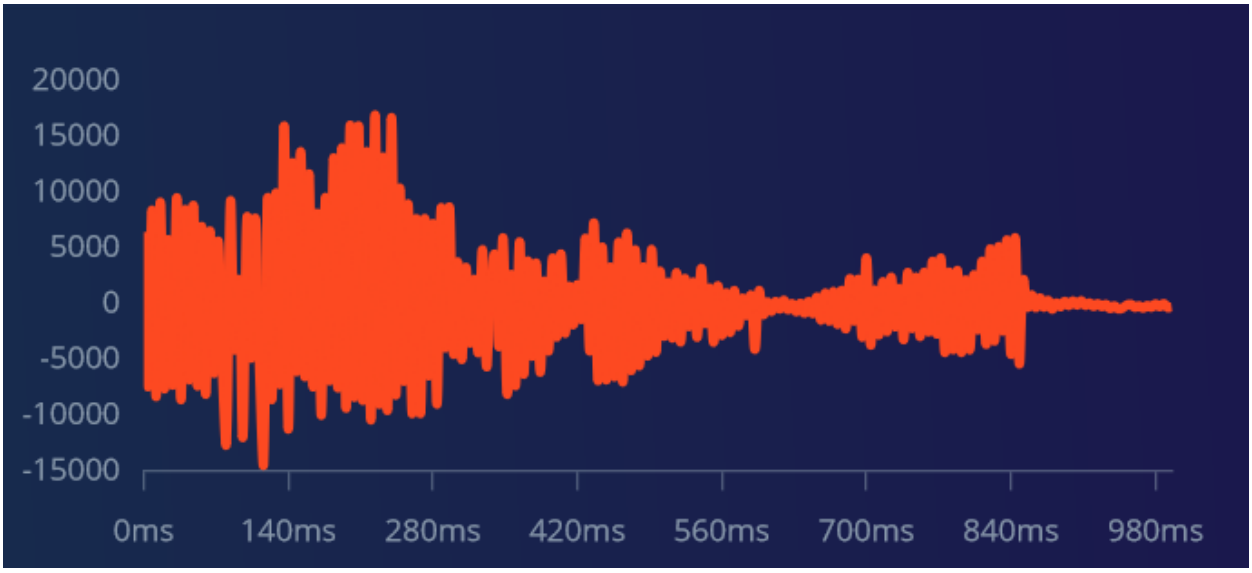
Sl. 3.2.1. Aruino Nano 33 BLE pločica

Također jedna od mogućnosti je korištenje bluetooth tehnologije što znači da se proširuje spektar opcija za izmjenu podataka s drugim uređajima, ali to ujedno i ograničava dostupnu energiju koja se može iskoristiti na druge opcije koje su ponuđene kao na primjer korištene senzora koji su prethodno ugrađeni. Arduino Nano 33 BLE je odličan izbor za takozvani „embodied machine learning“ jer nudi jako puno opcija i ima dovoljno memorije i brzine za projekt.

Razvojna pločica Arduino Nano sadrži brojne senzore kao što su senzor za temperaturu, barometar, senzor pokreta, senzor za tlak. U sebi sadrži MP34DT05 mikrofon senzor namijenjen prikupljanju zvuka iz okoline. Prikladna jemza prikupljanje podataka i implementaciju modela strojnog učenja pomoću TinyML modela.

3.3. Audio signali

Jedan od najrasprostranjenijih oblika prijenosa informacija su audio signali, koji su ukomponirani kao dio većine tehnologija oko nas. Audio signali su analogni ili digitalni električni signali koji prenose zvuk preko zvučnika ili slušalica. Ti zvukovi mogu biti različitih oblika, uključujući govor, glazbu, zvukove okoline i druge zvukove koji se pojavljuju u svakodnevnom životu. Svaki audio signal može se opisati različitim svojstvima, kao što su amplituda (jačina zvuka), frekvencija (tonalitet), trajanje, oblik vala i mnogi drugi. Analiza i obrada audio signala omogućuju nam razumijevanje, manipulaciju i interpretaciju zvukova, što je ključno za razvoj raznih tehnologija, uključujući prepoznavanje govora, analizu glazbe, telekomunikacije, nadzor i sigurnost, medicinsku dijagnostiku i razne druge primjene.Zvučni signal predstavlja kombinaciju različitih frekvencijskih komponenti, a frekvencija se izražava u jedinici Hertz (Hz) [2]. Poput svakog vala, zvuk nastaje zbog vibrirajućih objekata kao što su žice gitare ili dijafragma bubnja, što uzrokuje promjene u zraku. U ljudskom govoru, te promjene nastaju zbog vibracija naših glasnica. Zvuk se može čuti ako je frekvencija vibracija između 20 Hz i 20 kHz. Temeljna frekvencija zvuka naziva se ton. Visoki tonovi odgovaraju visokim frekvencijama, dok niski tonovi odgovaraju niskim frekvencijama. Zvučni signal može sadržavati više frekvencijskih komponenti, a raspon tih frekvencija naziva se spektrom. Glasnoća zvučnog signala na različitim frekvencijskim komponentama mjeri se u apsolutnoj snazi ili na decibelnoj skali. Prikaz audio signala u vremenskoj domeni na slici *Sl. 3.3.1.* dočarava kako se mali zvučni isječak prikazuje u digitalnom obliku.



Sl. 3.3.1. Prikaz audio signala u vremenskoj domeni

3.4. Platforme i alati

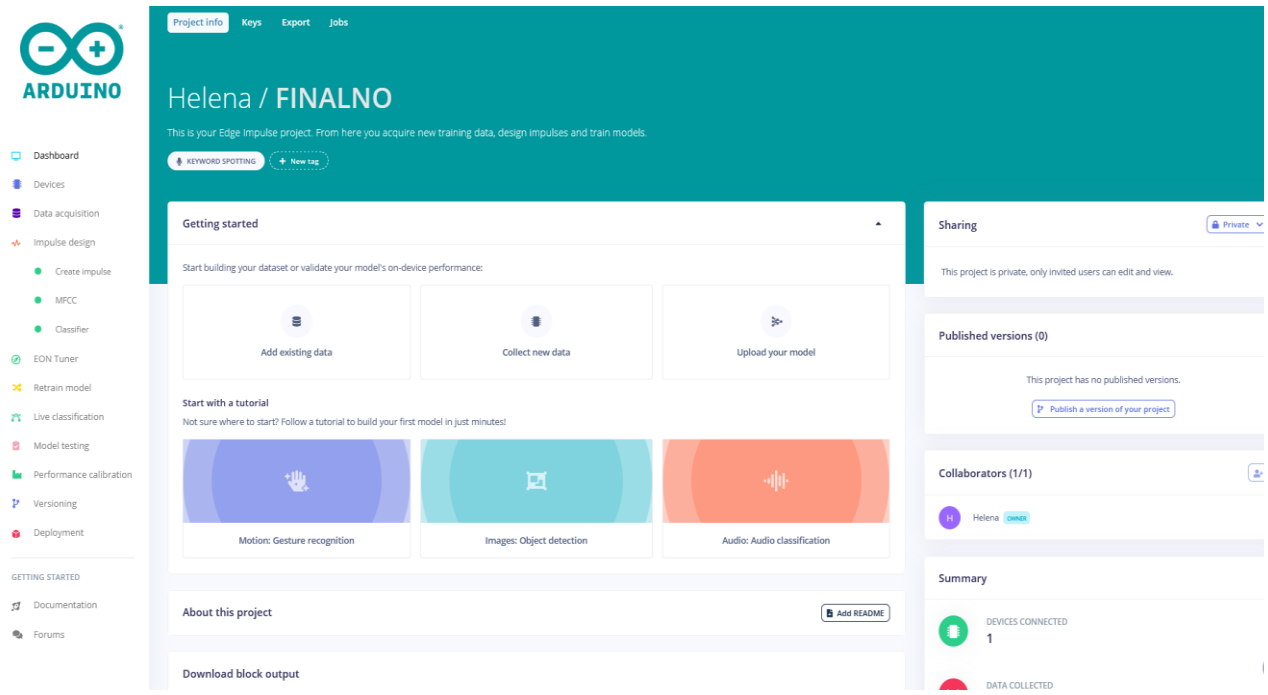
Napretkom i razvojem tehnologije ponuđene su nam optimalne platforme strojnog učenja koje možemo primjeniti na mikroupravljačke sustave. U ovome dijelu predstavljaju se platforme koje su pogodne za strojno učenje prema svojim pozitivnim i negativnim stranama.Neke platforme koje se mogu koristiti na mikroupravljačima su tinyML, EML, Micropython, Zephyr, Edge impulse. Također postoje mnoge druge platforme koje su prikladne za implementaciju modela strojnog učenja na mikroupravljače jer ovo područje konstantno evoluira.

3.4.1. Edge impulse

Edge Impulse platforma je jedna od pogodnijih platformi za rad zbog svoje prilagodbe i mogućnosti implementacije na brojne mikroupravljače i naravno koristi ju velik broj korisnika. Platforma nudi kompleksno rješenje koje olakšava proces stvaranja, treniranja i implementacije modela strojnog učenja na uređajima s ograničenim resursima. Kroz intuitivno sučelje, korisnici mogu brzo učitati podatke, označiti ih, provesti analizu i treniranje modela, te implementirati ih direktno na ciljane uređaje. Ova platforma omogućuje inženjerima da efikasno iskoriste potencijal senzorskih podataka, otvarajući vrata novim inovacijama u područjima kao što su pametni gradovi, zdravstvo, industrija i mnogi drugi sektori.

Platforma prikazana na slici pruža korisnički prijateljsko sučelje koje vodi korisnike kroz cijeli proces razvoja modela strojnog učenja [5]. To uključuje korištenje raznih optimiziranih blokova za predobradu i učenje, različitih arhitektura neuronskih mreža te unaprijed treniranih modela. Osim toga, Edge Impulse može generirati gotove binarne datoteke koje su spremne za implementaciju na stvarne uređaje, omogućujući jednostavno testiranje i implementaciju modela.

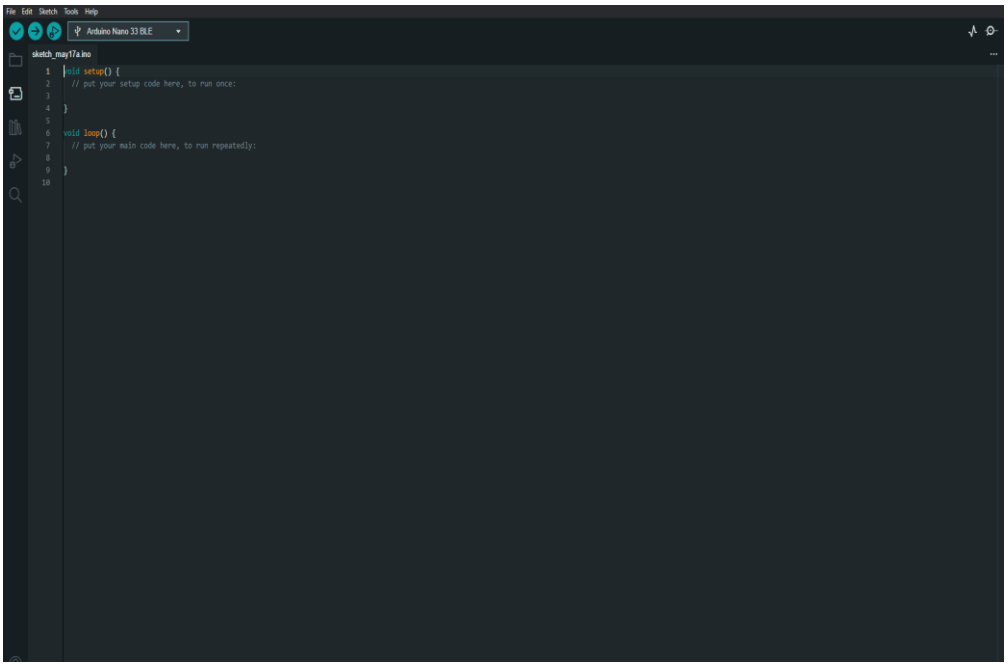
Edge Impulse optimizira modele strojnog učenja za izvršavanje izravno na rubnim uređajima (*engl. Edge device*).Rubni uređaji obrađuju podatke lokalno, na rubu mreže, umjesto da ih šalju u oblak ili podatkovni centar. To omogućava bržu analizu i reakciju u stvarnom vremenu, kao i povećanu sigurnost podataka. Primjeri uključuju pametne senzore u industriji, autonomna vozila, pametne kućanske uređaje [6]. Ova značajka omogućuje nisku latenciju i obradu u stvarnom vremenu, što je ključno za aplikacije koje zahtijevaju brzu reakciju na podatke iz senzora. Korištenjem rubnog računarstva, uređaji mogu donositi odluke i vršiti analize bez potrebe za stalnom vezom sa oblakom (*engl. cloud*), čime se povećava autonomija i smanjuju troškovi prijenosa podataka.Također platforma nudi podršku za širok spektar senzora, uključujući akcelerometre, mikrofone i kamere. Ova raznolikost čini platformu svestranom i primjenjivom u različitim projektima. Bez obzira na specifične potrebe projekta, omogućuje integraciju različitih vrsta senzora kako bi se ostvarili specifični ciljevi prepoznavanja i analize podataka.



Sl. 3.4.1. Edge Impulse platforma

3.4.2. Arduino razvojno okruženje

Arduino platforma je otvorena hardverska i softverska platforma koja omogućuje izradu i programiranje različitih digitalnih uređaja, uključujući mikrokontrolere, senzore, motorne pogone i druge zanimljive projekte. Ova platforma je posebno popularna među elektroničarima, studentima, inženjerima i hobistima zbog svoje jednostavnosti korištenja i fleksibilnosti. Arduino koristi različite vrste mikrokontrolera, uključujući popularne modele poput Arduino Uno, Arduino Nano, Arduino Mega, i drugih. Ovi mikrokontroleri obično imaju različite brojeve digitalnih i analognih ulaza/izlaza, što omogućuje raznolike projekte. Za programiranje Arduino uređaja koristi se Arduino Integrirano razvojno okruženje (*engl.Arduino Integrated Development Environment*) ili skraćeno Arduino IDE prikazan na *Sl. 3.4.2.1*. Besplatan softverski alat dostupan za različite platforme kao što su Windows, macOS, Linux. Arduino IDE omogućuje jednostavno pisanje, kompajliranje i prenošenje programa na Arduino uređaje. Arduino koristi jednostavan programski jezik temeljen na C/C++. Arduino IDE dolazi s ugrađenim bibliotekama i funkcijama koje olakšavaju interakciju s različitim perifernim uređajima poput senzora, ekrana ili motora. Arduino se koristi u različitim područjima, uključujući robotiku, Internet stvari (*engl. Internet of Things*), umjetnu inteligenciju, automatizaciju, digitalnu umjetnost i mnoge druge. Zbog svoje pristupačnosti i fleksibilnosti, Arduino platforma je popularna u različitim područjima. Postoji velika zajednica Arduino korisnika i entuzijasta koji dijele svoja iskustva, projekte i rješenja putem interneta. Izuzetno pogodna platforma za jednostavnu implementaciju i pokretanje jednostavnih modela u stvarnom vremenu.



Sl. 3.4.2.1. Prikaz Arduno IDE okruženja

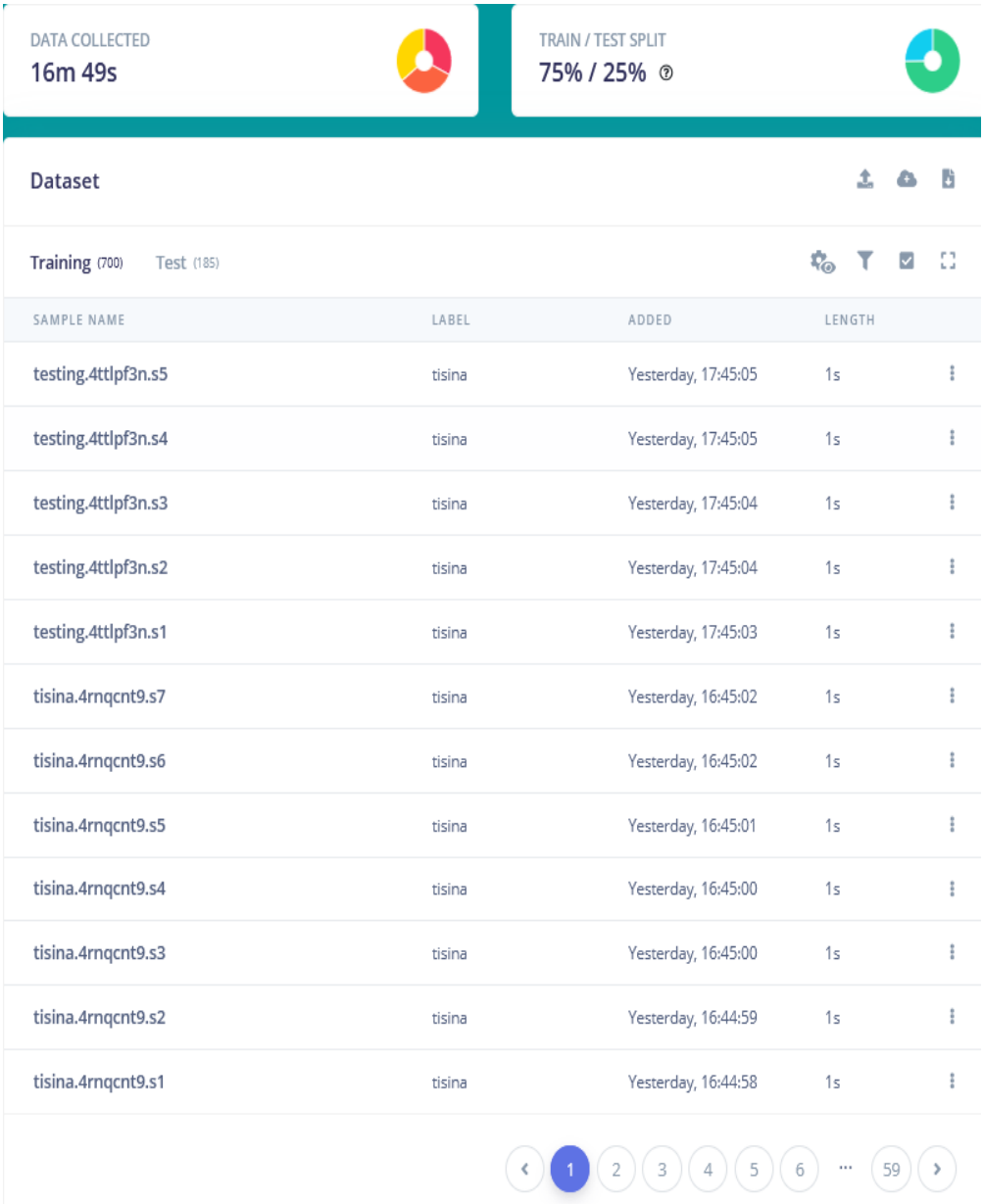
4. PRAKTIČNI DIO RADA

4.1. Prikupljanje podataka

Za pripremu podataka, audio signali su snimani korištenjem ugrađenog mikrofona na Arduino Nano Sense BLE mikroupravljačkoj platformi. Snimci su zatim preneseni na Edge Impulse platformu, gdje je proveden proces obrade podataka. Svaki snimak je podijeljen na dijelove duljine 1 sekunde, kako bi se izdvojili relevantni segmenti u kojima su izgovorene ključne riječi "crvena" i "plava", te segmenti u kojima je zabilježeno stanje tišine, odnosno kada se nisu izgovarale ključne riječi. Kako bi podatci bili što raznolikiji audio je sniman na raznim udaljenostima od mikrofona. Prikupljeni set podataka sniman je u prostoriji koja ima jako nisku razinu buke, nema pojave odzvanjanja to jest odjeka. Ovaj proces omogućuje stvaranje skupa podataka koji se može koristiti za treniranje klasifikatora za prepoznavanje izgovorenih riječi, kao i za detekciju tišine između njih.

Audio podatci koji su skupljeni razvrstani su na tri dijela „crvena“, „plava“ i „tišina“. Ukupna vremenska duljima podataka koji su prikupljeni je 16 minuta i 49 sekunda što je dovoljna količina za zadovoljavajuć rad i trening modela. Podatci za trening izrezani su tako da svaki izgovor riječi „crvena“ ili „plava“ traje 1 sekundu kao i „tišina“ koja predstavlja sve što nisu izgovorene ključne riječi.

Na slici *Sl. 3.3.1.*prikazani su prikupljeni audio podatci koji su pohranjeni na Edge Impulse platformu. Prikazana je podjela isječenih audio podatka i raspodjela između seta namijenjenog za treniranje i testiranje. Optimalna raspodjela podataka je 75% materijala za trening i 25% za testiranje.



Sl. 3.3.1. Prikaz prikupljenih audio podataka

4.2. Odabir modela

Pri odabiru modela uzima se u obzir baza podataka koja se koristi, u ovom slučaju koristi se baza snimljenih audio podataka, mora se gledati efikasnost obavljanja zadanog zadatka to jest koja opcija je najbliža rješavanju zadanog problema. Kao primjer prikazivanja učinkovitosti strojnog učenja koriste audio snimke ljudskoga glasa u obzir dolaze dvije opcije izabiranja procesorskog bloka, a te opcije su MFCC i MFE. Oba bloka su namijenjena korištenju pri obradi zvuka.

Mel-Frekvencijski Kepstralni Koeficijenti (*engl. Mel-frequency cepstral coefficients*) ili MFCC predstavljaju temeljni alat u analizi zvuka i prepoznavanju govora, a njihova primjena na mikroupravljačkim platformama otvara vrata novim mogućnostima u području ugrađenih sustava. Ova tehnika, koja je potekla iz područja obrade govornih signala, pokazala se izuzetno efikasnom u ekstrakciji ključnih značajki zvuka,.Procesorski blok MFE (*engl. Mel-filterbank energies*) za obradu zvuka izvlači vremenske i frekvencijske značajke iz signala [4]. Međutim, koristi nelinearnu skalu u frekvencijskom području, nazvanu Mel-skala. Pokazuje dobre performanse na audio podacima, uglavnom za slučajeve klasifikacije zvuka kada zvukovi koje treba klasificirati mogu biti razlikovani ljudskim sluhom, ali ne uključuju prepoznavanje glasa.

Prva ključna karakteristika MFCC je efikasnost u obradi. Na mikroupravljačkim platformama, koje su često ograničene resursima poput procesorske snage i memorije, važno je koristiti tehnike koje zahtijevaju minimalne resurse. MFCC pružaju kompaktnu reprezentaciju zvuka koja zahtijeva relativno malo resursa za obradu, čineći ih idealnim izborom za mikroupravljače s niskom potrošnjom energije. MFCC zahtijevaju samo male količine memorije za pohranu izračunatih značajki. Na mikroupravljačkim platformama, gdje je memorija često ograničen resurs, ova karakteristika postaje ključna. Implementacija MFCC algoritama na mikroupravljačima ne uzrokuje značajno povećanje kompleksnosti ili troškova, što olakšava razvoj aplikacija u ovom području.

Prilagodljivost MFCC također je važna u kontekstu mikroupravljačkih platformi. Parametri poput broja koeficijenata i veličine okvira mogu se prilagoditi specifičnim potrebama i ograničenjima platforme, čime se optimizira učinkovitost i performanse algoritma. Ova prilagodljivost omogućuje razvoj aplikacija koje su prilagođene različitim mikroupravljačkim platformama i njihovim specifičnim zahtjevima.Konačno, jednostavna implementacija MFCC algoritama čini ih pristupačnim čak i osobama s ograničenim iskustvom u obradi signala. Postoji mnogo dostupnih

biblioteka i resursa koji olakšavaju implementaciju MFCC na mikroupravljačkim platformama, što omogućuje brzi razvoj raznovrsnih aplikacija u području ugrađenih sustava.

U zaključku, Mel-Frekvencijski Kepstralni Koeficijenti predstavljaju izvanredan alat za analizu zvuka i prepoznavanje govora na mikroupravljačkim platformama. Njihova efikasnost, niski zahtjevi za memorijom, prilagodljivost i jednostavna implementacija čine ih nezamjenjivim alatom za razvoj pametnih uređaja.

Kao blok za učenje pogodan odabir je klasifikator. Osnovna ideja je da će klasifikator neuronske mreže uzeti neke ulazne podatke i dati vjerojatnosni rezultat koji pokazuje koliko je vjerojatno da ulazni podaci pripadaju određenoj klasi [7].

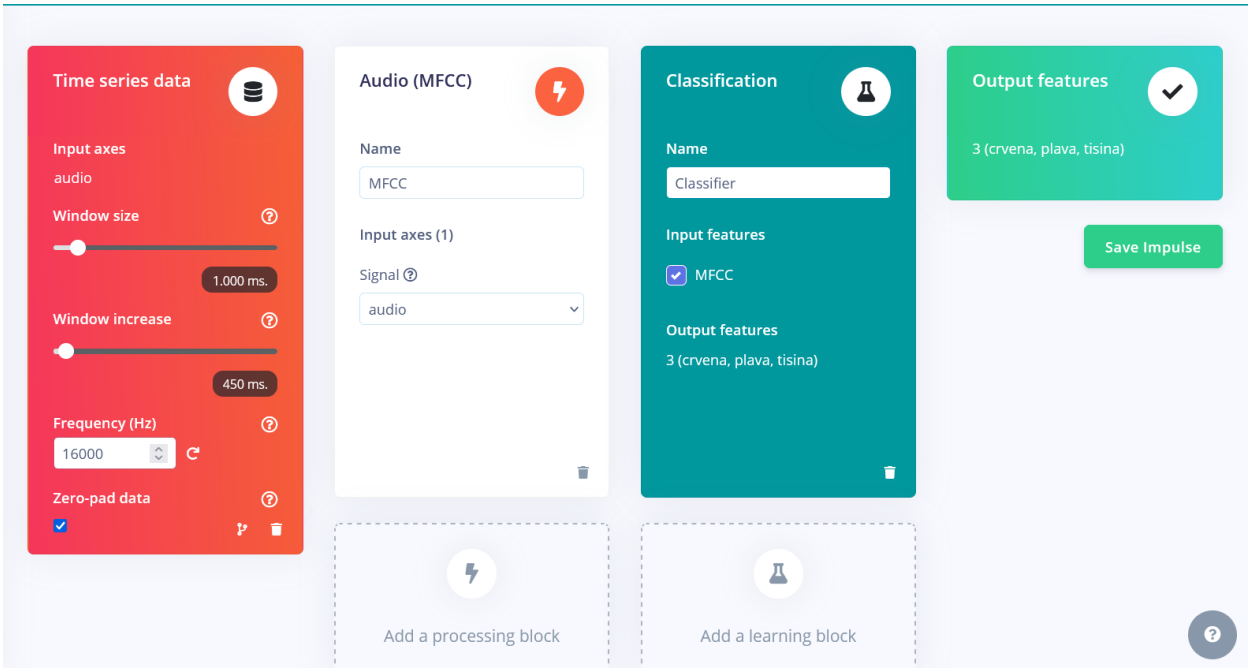
4.3. Treniranje i implementacija modela

Nakon što su podaci pripremljeni, slijedi faza dizajna i treniranja modela. Edge Impulse nudi razne unaprijed definirane blokove za predobradu i učenje koji se mogu prilagoditi specifičnim potrebama projekta. Korisnici mogu birati između različitih arhitektura neuronskih mreža, kao što su konvolucijske neuronske mreže (CNN) za prepoznavanje slika ili rekurentne neuronske mreže (RNN) za analizu vremenskih nizova.

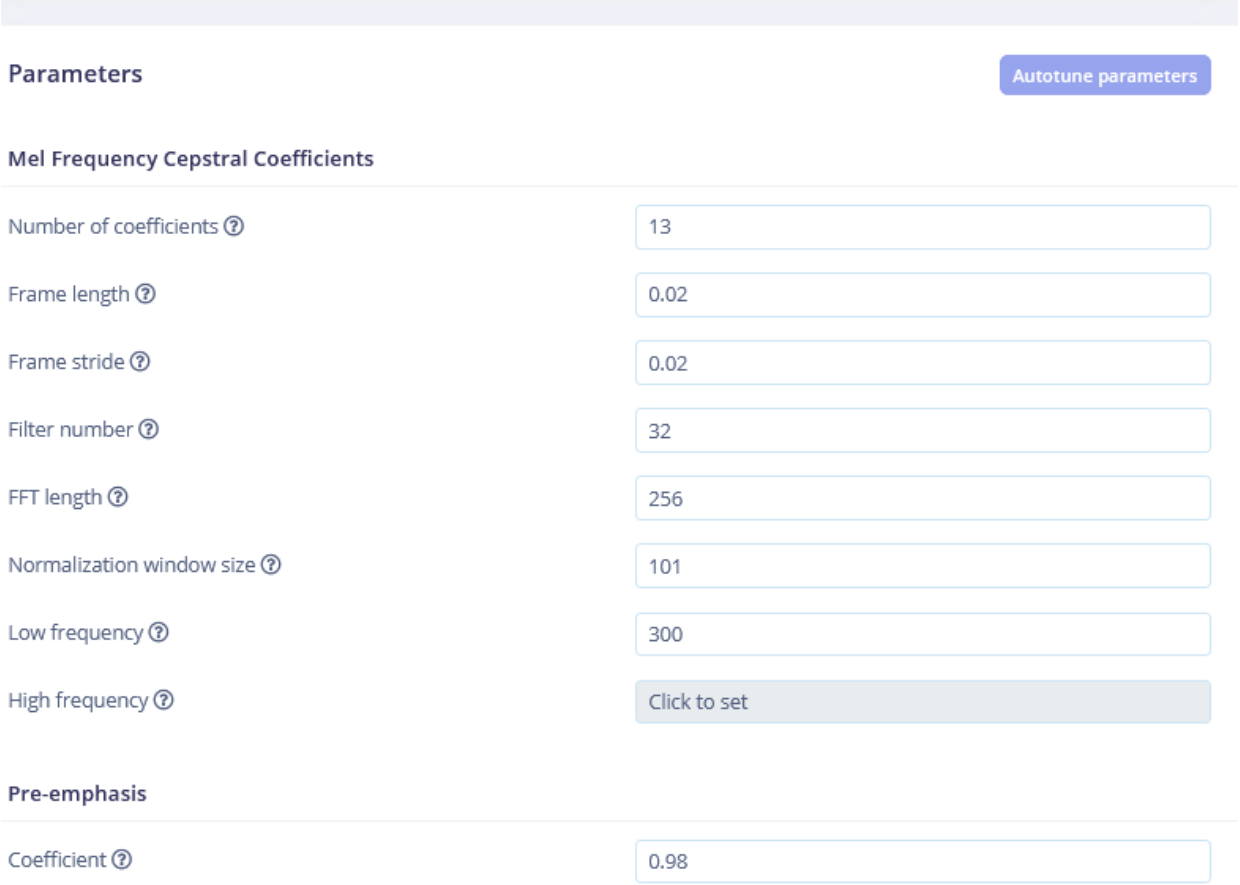
Proces treniranja modela uključuje iterativno prilagođavanje težina unutar mreže kako bi se minimizirala pogreška predikcije na skupu podataka za treniranje. Edge Impulse automatski upravlja ovim procesom, omogućujući korisnicima da prate napredak treniranja kroz grafičko sučelje.

Najoptimalniji način prilagodbe podataka je postaviti veličinu prozora na 1000 ms ili 1 sekundu, a pomak prozora ili njegovo povećanje na 450 ms ili 0.45 sekunda prikazan na *Sl. 4.3.1* unutar bloka vremenski niz podataka (*engl. Time series data*).

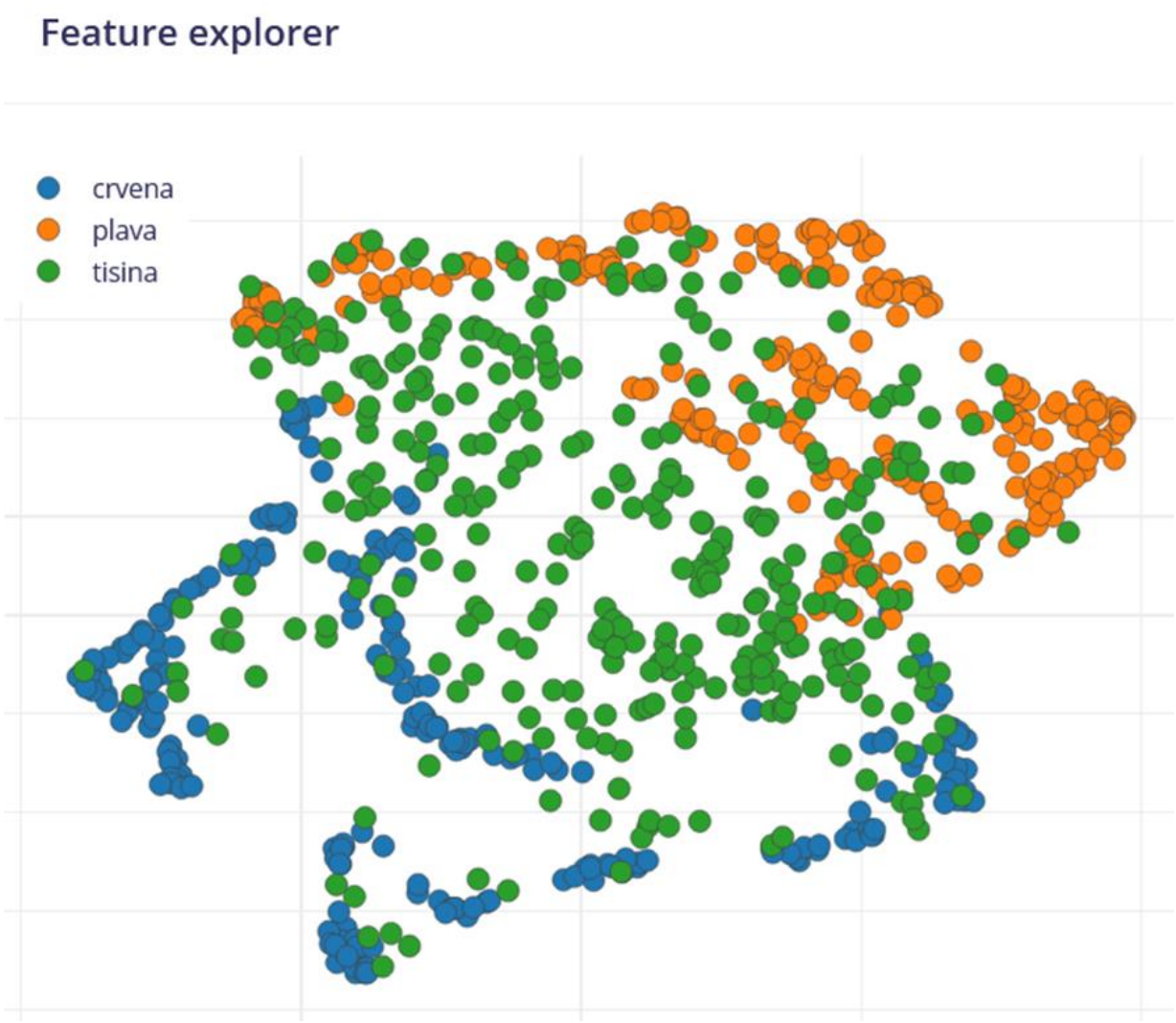
Parametri koji se modificiraju u MFCC bloku su broj koeficijenata, duljina okvira, korak okvira, broj filtera, duljina FFT-a, veličina prozora za normalizaciju, niska frekvencija, visoka frekvencija. Redom su postavljeni na vrijednosti 13, 0.02, 0.02, 32, 256,101,300, prazno prikazano na *Sl. 4.3.2*



Sl. 4.3.1 Prikaz odabira blokova

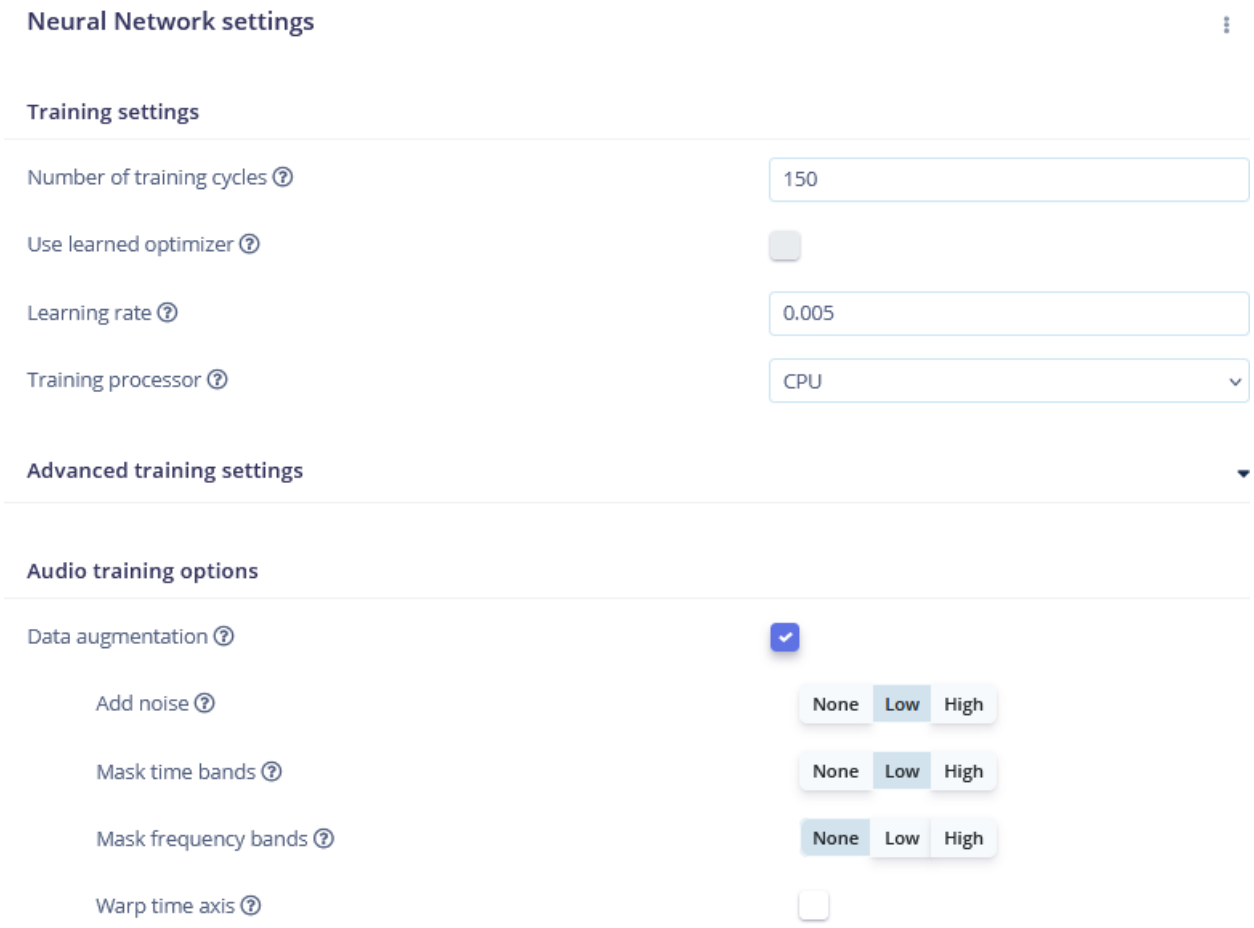


Sl. 4.3.2 Prikaz MFFC parametara



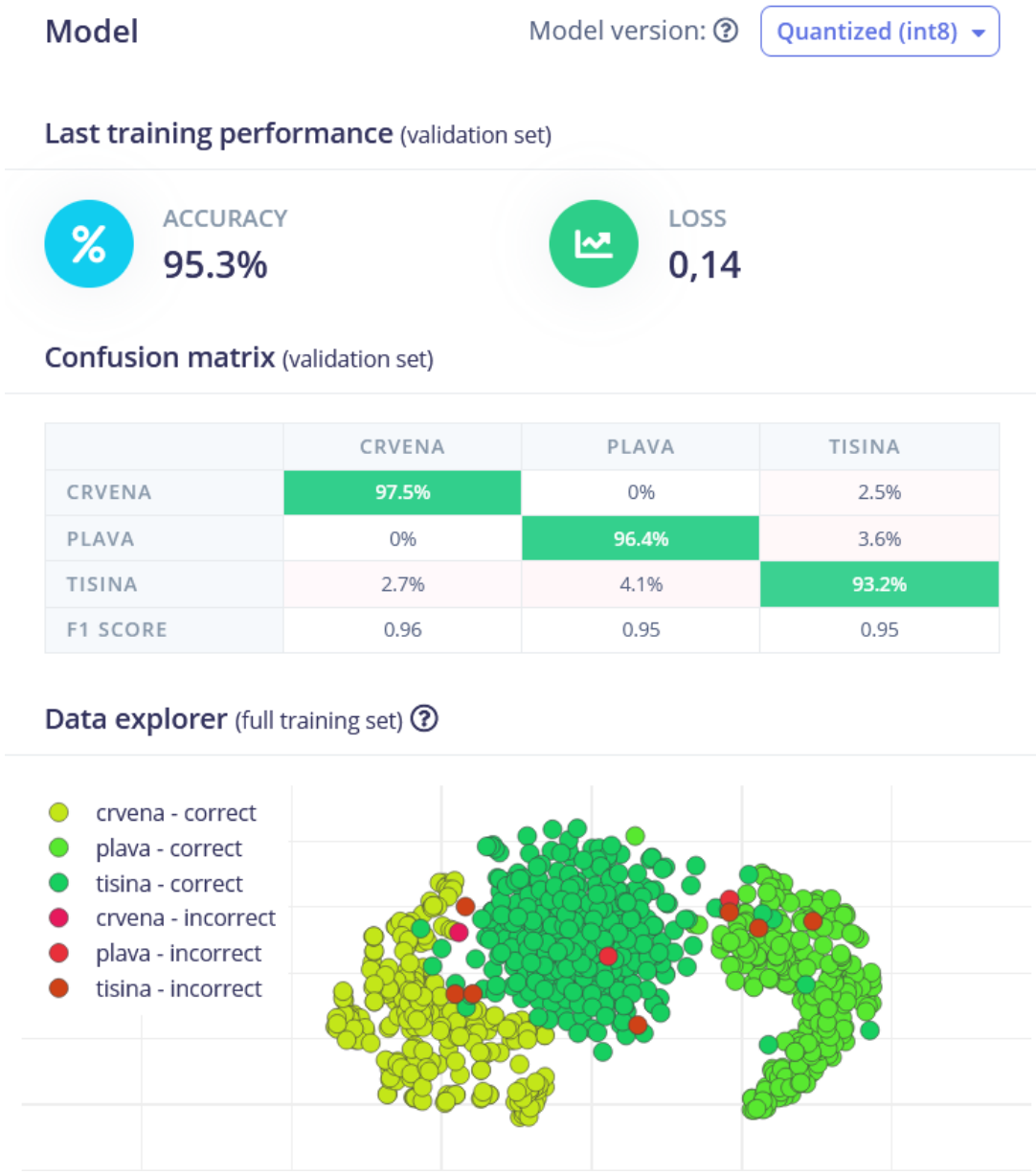
Sl. 4.3.3 Prikaz točnosti prepozavanja ključnih riječi

Na prikazanom grafu *Sl. 4.3.3* prikazani su podatci skupljeni koji predstavljaju izgovor „crvene“ i „plave“ i kako ih MFCC raspoznaje. Podatci su grupirani po bojama i to nam govori da je naš set podataka zadovoljavajuć. Ovaj alat nam pokazuje je li set podataka prikladan za trening i prije samog treninga. Alat korišten naziva se istraživač značajki (*engl. Feature explorer*), a njegova namjena je prikazati grupacije klasa prepoznatih ključnih riječi.



Sl. 4.3.4. Prikaz opcija za konfiguraciju neuronske mreže

Kako bi najoptimalnije prilagodili potreban model da prepoznaje ključne riječi moramo konfigurirati neuralnu mrežu. Model je konfiguriran za 150 ciklusa treniranja, sa stopom učenja koja kontrolira koliko se unutarnji parametri modela ažuriraju tijekom svakog koraka procesa treniranja. Ako mreža brzo preuči (*engl. overfitting*), stopa učenja se može smanjiti kako bi se postigla optimalna učinkovitost. Proces treniranja se odvija na CPU-u, što omogućava učinkovitu obradu podataka i prilagodbu modela. Na stranici NN Classifier, koja postaje dostupna nakon izdvajanja značajki iz DSP bloka, korisnik može konfigurirati model i proces treniranja te dobiti pregled performansi modela.



Sl. 4.3.5. Prikaz performansi modela

Prikaz modela na *Sl. 4.3.5.* i njegove uspješnosti obrade podataka, to jest klasifikacija ključnih riječi. Model je dobro prilagođen i dobro prepoznaje riječi „crvena“ i „plava“, te ih razlikuje od drugih zvukova. Prikazana je efikasnost modela u platformi, prije implementacije na arduino pločicu. Problem prepoznavanja nastaje ako neka riječ zvuči slično ključnoj riječi kao na primjer riječi koje se rimuju kao što su „plava“ i „krava“, „crvena“ i „drvena“. Kako bi se problem rješio bilo bi potrebno prikupiti još podataka, ali radi se o malome jednostavnome modelu koji dosta dobro izvršava svoju funkciju osim u ekstremnim slučajevima to nije potrebno jer zadovoljava potrebe zadanog zadatka.

4.4. Analiza koda

Prikazan kod je prilagođen implementaciju u arduino okruženje , a to znači da je implementiran na Arduno nano razvojnu pločicu. Kod je objašnjen u segmentima. Edge Impulse platforma je razvila svoj vlastiti kompailer pod nazivom EON Compailer koji svoje značajke preuzima iz TensorflowLite compailera te dodaje dodatne mogućnosti.

```
1:    #define EIDSP_QUANTIZE_FILTERBANK   0
2:    #define EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW 4
3:
4:    #include <PDM.h>
5:    #include <FINALNIO_inferencing.h>
6:
7:    #define RED     22
8:    #define BLUE    24
9:
10:   typedef struct {
11:       signed short *buffers[2];
12:       unsigned char buf_select;
13:       unsigned char buf_ready;
14:       unsigned int buf_count;
15:       unsigned int n_samples;
16:   } inference_t;
17:
18:   static inference_t inference;
19:   static bool record_ready = false;
20:   static signed short *sampleBuffer;
21:   static bool debug_nn = false; // Set this to true to see e.g. features
      generated from the raw signal
22:   static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);
23:
24:   /**
25:    * @brief      Arduino setup function
26:    */
```

Sl. 4.4.1. Postavljanje parametara i pozivanje biblioteka

Isječak koda na *Sl. 4.4.1.* postavlja potrebne konfiguracije za Arduino projekt koji koristi Edge Impulse model za izvođenje audio inferencije. Definira makroe za određene konfiguracije, uključuje potrebne biblioteke, postavlja definicije pinova za LED-ice, te definira i inicijalizira varijable i strukture potrebne za upravljanje audio podacima i izvođenje inferencije. Struktura „inference_t“ je namijenjena je rukovanju sa audio spremnicima i praćenju prikupljanja podataka te njihovo procesiranje.

```
27:  void setup()
28:  {   pinMode(RED, OUTPUT);
29:      pinMode(BLUE, OUTPUT);
30:      // put your setup code here, to run once:
31:      Serial.begin(115200);
32:      // comment out the below line to cancel the wait for USB connection
     (needed for native USB)
33:      while (!Serial);
34:      Serial.println("Edge Impulse Inferencing Demo");
35:
36:      // summary of inferencing settings (from model_metadata.h)
37:      ei_printf("Inferencing settings:\n");
38:      ei_printf("\tInterval:               %.2f               ms.\n",
     (float)EI_CLASSIFIER_INTERVAL_MS);
39:      ei_printf("\tFrame                  size:                  %d\n",
     EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
40:      ei_printf("\tSample          length:          %d          ms.\n",
     EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
41:      ei_printf("\tNo.           of           classes:           %d\n",
     sizeof(ei_classifier_inferencing_categories) /
42:
     sizeof(ei_classifier_inferencing_categories[0]));
43:
44:      run_classifier_init();
45:      if    (microphone_inference_start(EI_CLASSIFIER_SLICE_SIZE)    ==
     false) {
46:          ei_printf("ERR: Could not allocate audio buffer (size %d),
     this  could  be  due  to  the  window  length  of  your  model\r\n",
     EI_CLASSIFIER_RAW_SAMPLE_COUNT);
47:          return;
48:      }
49:  }
50:
```

Sl. 4.4.2. Prikaz setup funkcije

Prikazani isječak *Sl. 4.4.2* koda predstavlja setup funkciju koja se koristi za inicijalno postavljanje potrebnih parametara i konfiguracija na Arduino pločici. Na početku, definiraju se pinovi za LED-ice, gdje se crvena (*engl. Red*) i plava (*engl. Blue*) postavljaju kao izlazni pinovi, omogućujući kontrolu LED lampica. Serijska komunikacija se inicijalizira na brzini od 115200 bita u sekundi, što omogućava komunikaciju Arduino pločice i računala. Nakon toga, ispisuju se osnovne informacije o postavkama modela za prepoznavanje ključnih riječi, uključujući interval uzorkovanja, veličinu okvira i broj klasifikacijskih kategorija. Slijedi inicijalizacija klasifikatora i pokretanje procesa prikupljanja audio podataka putem mikrofona. Ako inicijalizacija ne uspije zbog nedostatka memorije ili drugih problema, ispisuje se poruka o grešci koja informira korisnika o potencijalnim uzrocima.

Funkcija uključuje postavljanje pinova za LED-ice, inicijalizaciju serijske komunikacije, ispisivanje sažetka postavki inferencije, inicijalizaciju klasifikatora te pokreće prikupljanje podataka pomoću mikrofona, te ako inicijalizacija ne uspije, ispisuje se odgovarajuća poruka o grešci.

```
54:   void loop()
55:   {
56:       bool m = microphone_inference_record();
57:       if (!m) {
58:           ei_printf("ERR: Failed to record audio...\n");
59:           return;
60:       }
61:
62:       signal_t signal;
63:       signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
64:       signal.get_data = &microphone_audio_signal_get_data;
65:       ei_impulse_result_t result = {0};
66:
67:       EI_IMPULSE_ERROR r = run_classifier_continuous(&signal, &result,
      debug_nn);
68:       if (r != EI_IMPULSE_OK) {
69:           ei_printf("ERR: Failed to run classifier (%d)\n", r);
70:           return;
71:       }
72:
73:    if (result.classification[0].value >= 0.85) {
74:       digitalWrite(BLUE, HIGH);
75:       digitalWrite(RED, LOW);
76:
77:     }
78:         else if (result.classification[1].value >= 0.85) {
79:       digitalWrite(BLUE, LOW);
80:       digitalWrite(RED, HIGH);
81:
82:     }
```

Sl.4.4.4. Funkcija kontinuirane audio klasifikacija s kontrolom paljenja LED dioda a)

```
83.       if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW))
      {
84.           // print the predictions
85.           ei_printf("Predictions ");
86.           ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d
      ms.)",
87.               result.timing.dsp, result.timing.classification,
      result.timing.anomaly);
88.           ei_printf(": \n");
89.           for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
90.               ei_printf("    %s: %.5f\n",
      result.classification[ix].label,
91.                         result.classification[ix].value);
92.           }
93.   #if EI_CLASSIFIER_HAS_ANOMALY == 1
94.           ei_printf("    anomaly score: %.3f\n", result.anomaly);
95.   #endif
96.
97.           print_results = 0;
98.       }
99.   }
100.
101.  /**
102.   * @brief      PDM buffer full callback
103.   *             Get data and call audio thread callback
104.   */
```

Sl. 4.4.5. Funkcija kontinuirane audio klasifikacija s kontrolom paljenja LED dioda b)

Funkcija „loop()“ prikazana na slikama *Sl.4.4.4.* i *Sl. 4.4.5.* kontinuirano snima audio podatke, obrađuje ih putem klasifikatora i upravlja LED diodama na temelju rezultata klasifikacije. Prvo, poziva se funkcija „microphone_inference_record()“ koja snima audio podatke s veličinom prozora od 1000 ms i povećanjem prozora od 450 ms. Vjerojatnost da će se snimiti samo dio izgovorene riječi je značajno smanjena. Preklapanje prozora osigurava da će cijele riječi gotovo uvijek biti unutar barem jednog od prozora, a često i unutar dva uzastopna prozora, čime se poboljšava točnost prepoznavanja i klasifikacije riječi. U slučaju da snimanje ne uspije, ispisuje se poruka o grešci i funkcija završava. Zatim se pokreće funkcija „run_classifier_continuous“ koja kontinuirano obrađuje snimljene audio podatke pomoću klasifikatora.

Ako klasifikacija ne uspije, ispisuje se poruka o grešci i funkcija završava. Na kraju, funkcija provjerava rezultate klasifikacije: ako je vjerojatnost prve klase (result.classification[0].value) veća ili jednaka 0.85, plava LED dioda se uključuje, a crvena isključuje; ako je vjerojatnost druge klase (result.classification[1].value) veća ili jednaka 0.85, crvena LED dioda se uključuje, a plava isključuje. Tako funkcija `loop()` kontinuirano snima audio podatke, klasificira ih i upravlja LED diodama na temelju rezultata klasifikacije

Funkcija loop() ne samo da kontinuirano snima i obrađuje audio podatke te upravlja LED diodama na temelju rezultata klasifikacije, već i periodično ispisuje rezultate klasifikacije. Nakon provjere i upravljanja LED diodama, dodan je segment koda koji osigurava periodičan ispis predikcija klasifikatora. Ako je broj iteracija „print_results“ jednak ili veći od vrijednosti „EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW“, ispisuju se predikcije, uključujući informacije o vremenu digitalne obrade signala, klasifikacije i anomalija. Zatim se ispisuju vrijednosti svih klasifikacijskih oznaka i njihove pripadajuće vjerojatnosti. Ako je detekcija anomalije omogućena, ispisuje se i rezultat za anomalije. Nakon ispisa, brojač „print_results“ se resetira na nulu. Ovaj kod omogućuje korisniku uvid u performanse modela u stvarnom vremenu, pružajući korisne informacije o rezultatima klasifikacije.

```
105:  static void pdm_data_ready_inference_callback(void)
106:  {
107:      int bytesAvailable = PDM.available();
108:
109:          int    bytesRead    =    PDM.read((char    *)&sampleBuffer[0],
      bytesAvailable);
110:
111:      if (record_ready == true) {
112:          for (int i = 0; i<bytesRead>> 1; i++) {
113:
      inference.buffers[inference.buf_select][inference.buf_count++]      =
      sampleBuffer[i];
114:
115:              if (inference.buf_count >= inference.n_samples) {
116:                  inference.buf_select ^= 1;
117:                  inference.buf_count = 0;
118:                  inference.buf_ready = 1;
119:              }
120:          }
121:      }
122:  }
123:
124:  /**
125:   * @brief      Init inferencing struct and setup/start PDM
126:   *
127:   * @param[in]  n_samples  The n samples
128:   *
129:   * @return     { description_of_the_return_value }
       */
```

Sl.4.4.6. PMD funkcija povratnog poziva za spremanje audio podataka

Kod prikazan na slici *Sl.4.4.6.* definira funkciju „pdm_data_ready_inference_callback“, koja se aktivira kada je PDM spremnik spreman za obradu podataka. Prvo, funkcija provjerava koliko je bajtova dostupno za čitanje iz PDM spremnika, a zatim čita te podatke u „sampleBuffer“. Ako je snimanje audio podataka spremno (označeno s „record_ready“), podaci se prenose u odgovarajući spremnik za obradu. Kad je spremnik pun, mijenja se aktivni spremnik i označava da je spreman za obradu. Ova funkcija omogućuje kontinuirano snimanje i obradu audio podataka za daljnju analizu i klasifikaciju.

```
130:  static bool microphone_inference_start(uint32_t n_samples)
131:  {
132:  inference.buffers[0] = (signed short *)malloc(n_samples *
      sizeof(signed short));
133:
134:  if (inference.buffers[0] == NULL) {
135:  return false;
136:  }
137:
138:  inference.buffers[1] = (signed short *)malloc(n_samples *
      sizeof(signed short));
139:
140:  if (inference.buffers[1] == NULL) {
141:  free(inference.buffers[0]);
142:  return false;
143:  }
144:
145:  sampleBuffer = (signed short *)malloc((n_samples >> 1) *
      sizeof(signed short));
146:
147:  if (sampleBuffer == NULL) {
148:  free(inference.buffers[0]);
149:  free(inference.buffers[1]);
150:  return false;
151:  }
152:
```

Sl.4.4.7. Funkcija za postavljanje PMD mikrofona za audio interferenciju a)

```
153:
154:      inference.buf_select = 0;
155:      inference.buf_count = 0;
156:      inference.n_samples = n_samples;
157:      inference.buf_ready = 0;
158:
159:
160:      PDM.onReceive(&pdm_data_ready_inference_callback);
161:
162:      PDM.setBufferSize((n_samples >> 1) * sizeof(int16_t));
163:
164:
165:      if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
166:          ei_printf("Failed to start PDM!");
167:      }
168:
169:      // set the gain, defaults to 20
170:      PDM.setGain(127);
171:
172:      record_ready = true;
173:
174:      return true;
175:  }
176:
177:  /**
178:   * @brief      Wait on new data
179:   *
180:   * @return     True when finished
181:   */
```

Sl.4.4.8. Funkcija za postavljanje PMD mikrofona za audio interferenciju b)

Funkcija „microphone_inference_start“, prikazana na slikama *Sl.4.4.7.* i *Sl.4.4.8.* odgovorna je za početak prikupljanja audio podataka pomoću mikrofona i pripremu potrebnih struktura i resursa za obradu tih podataka.

```
182:  static bool microphone_inference_record(void)
183:  {
184:      bool ret = true;
185:
186:      if (inference.buf_ready == 1) {
187:          ei_printf(
188:              "Error sample buffer overrun. Decrease the number of
      slices per model window "
189:              "(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW)\n");
190:          ret = false;
191:      }
192:
193:      while (inference.buf_ready == 0) {
194:          delay(1);
195:      }
196:
197:      inference.buf_ready = 0;
198:
199:      return ret;
200:  }
```

Sl. 4.4.9. Funkcija za upravljanje snimanjem podataka pomoću mikrofona

Funkcija „microphone_inference_record“ prikazana na slici *Sl. 4.4.9.* , provjerava je li spremnik uzoraka spreman za snimanje novih podataka. Ako je spremnik spreman, to znači da je prethodni set uzoraka već prethodno obrađen, a novi podaci se još nisu spremili. U tom slučaju, funkcija ispisuje poruku o prekoračenju spremnika uzoraka i vraća false, označavajući da je snimanje neuspješno.

Nakon toga, funkcija ulazi u petlju u kojoj čeka dok se spremnik ne označi kao spreman . Ova petlja osigurava da se funkcija ne završi prije nego što spremnik postane spreman za snimanje novih podataka. Kada spremnik postane spreman, postavlja se natrag na stanje "nije spreman" i funkcija vraća vrijednost ret, koja označava je li snimanje uspješno.

```
201: static int microphone_audio_signal_get_data(size_t offset, size_t
     length, float *out_ptr)
202: {
203:     numpy::int16_to_float(&inference.buffers[inference.buf_select ^
     1][offset], out_ptr, length);
204:
205:     return 0;
206: }
207:
208: /**
209:  * @brief      Stop PDM and release buffers
210:  */
```

Sl. 4.4.10. Funkcija za pretvaranje audio podataka iz spremnika u float oblik

Funkcija „microphone_audio_signal_get_data“ prikazana na slici *Sl. 4.4.10.* dobiva sirove audio podatke iz spremnika i pretvara ih u float vrijednosti. Prima tri argumenta: offset, koji označava mjesto u spremniku odakle počinjemo čitati podatke, `length`, koji označava koliko uzoraka trebamo pročitati, te out_ptr, pokazivač na mjesto gdje pohranjujemo pretvorene float vrijednosti.

Funkcija koristi „numpy::int16_to_float“ metodu za pretvorbu uzoraka iz formata „int16“ u format float. Uzorci se čitaju iz spremnika, počevši od lokacije određene offset vrijednošću, i čita se broj uzoraka određen length vrijednošću. Pretvoreni float uzorci se zatim pohranjuju na lokaciju određenu pokazivačem out_ptr. Ukratno ovaj dio koda odgovoran je za direktno prikupljanje podataka sa mikrofona uređaja.

```
211:  static void microphone_inference_end(void)
212:  {
213:      PDM.end();
214:      free(inference.buffers[0]);
215:      free(inference.buffers[1]);
216:      free(sampleBuffer);
217:  }
218:
219:  #if    !defined(EI_CLASSIFIER_SENSOR)||    EI_CLASSIFIER_SENSOR    !=
      EI_CLASSIFIER_SENSOR_MICROPHONE
220:  #error "Invalid model for current sensor."
221:  #endif
```

Sl. 4.4.11. Funkcija za zaustavljanje PMD mikrofona i alokacije memorije spremnika

Funkcija „microphone_inference_end“ prikazana na *Sl. 4.4.11.*, završava snimanje audio podataka i oslobađa resurse koji su alocirani za pohranu uzoraka. Prvo, funkcija zaustavlja PDM mikrofon pozivom funkcije PDM.end(). Zatim oslobađa memoriju koja je alocirana za spremnike audio uzoraka (buffers[0] i buffers[1]) te za privremeni spremnik „sampleBuffer“ pomoću funkcije „free()“.

Nakon toga, koristi se preprocesorska direktiva #if kako bi se provjerilo je li model definiran za uporabu s mikrofonom kao senzorom. Ako model nije prikladan za trenutni senzor (u ovom slučaju mikrofon), generira se greška koja sprječava kompilaciju koda.

5. TESTIRANJE

Implementacijom napravljenog modela za prepoznavanje ključnih riječi „crvena“ i„plava“ na Arduino Nano 33 BLE mikroupravljač testira se koliko je učinkovit u raspoznavanju ključnih riječi i ostalih zvukova i riječi koji to nisu, a svrstani su pod kategoriju „tišina“.

Kako bi se procijenila točnost modela za prepoznavanje govora, provedeno je testiranje s više osoba. Testiranje je provedeno sa šest osoba, pri čemu je svaka osoba 100 puta izgovorila riječi „crvena“ i „plava“, te naravno izgovarane su i druge nevezane riječi koje se trebaju klasificirati kao kategorija „tišina“.

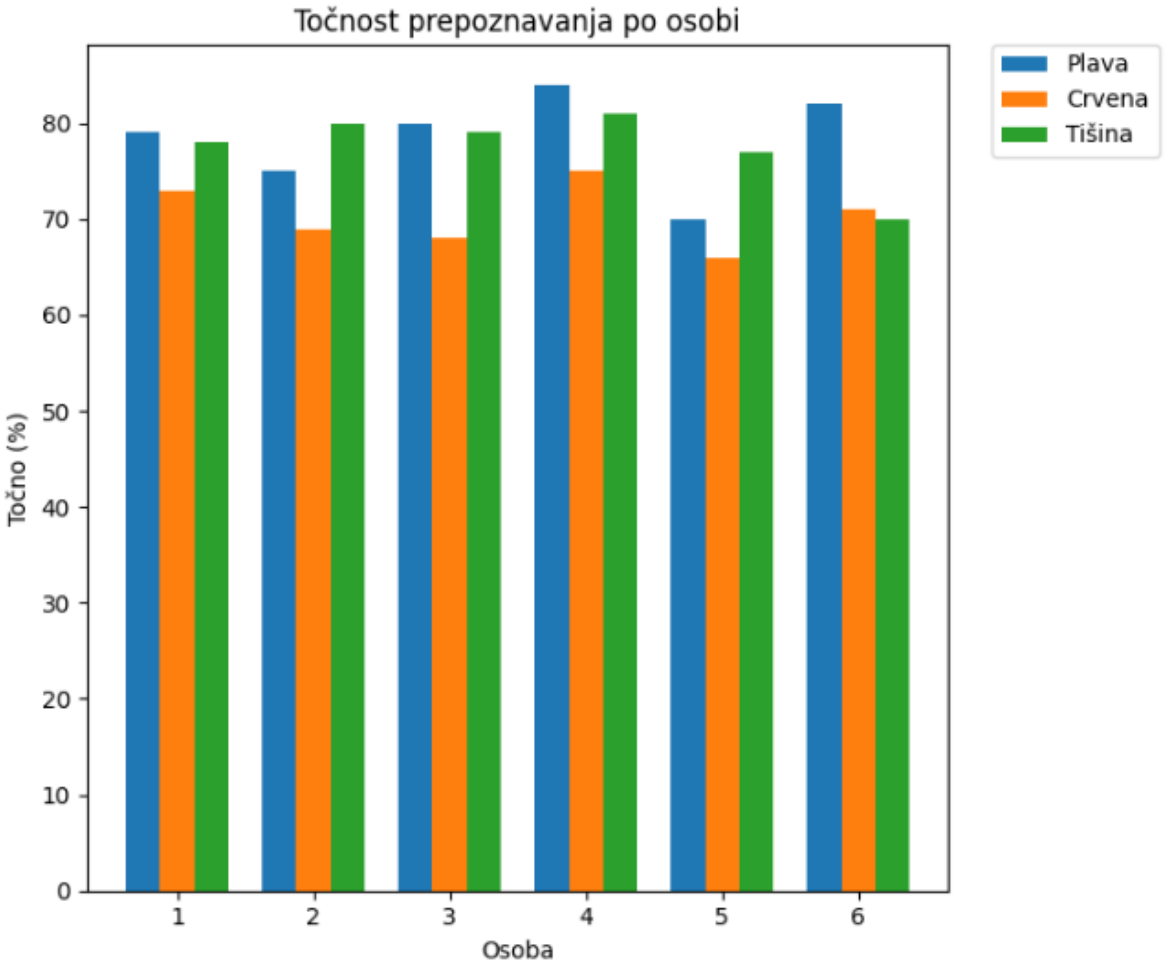
Svaki izgovor je zabilježen i analiziran pomoću modela za prepoznavanje govora. Rezultati su podijeljeni na točna i netočna prepoznavanja za svaku riječ.

U tablici *5.1.* prikazani su podatci testiranja modela direktno na arduino mikroupravljaču. Testiranje je obavljeno u uvjetima gdje su vanjski šum i zvukovi minimalni.

Tablica 5.1. Podatci o efikasnosti modela

Osoba	Ključna riječl	Točni	Netočno	Točnost (%)
1	Plava	75	25	75%
	Crvena	73	27	73%
	Tišina	70	30	70%
2	Plava	75	25	75%
	Crvena	69	31	69%
	Tišina	80	20	80%
3	Plava	80	20	80%
	Crvena	68	32	68%
	Tišina	79	21	79%
4	Plava	84	16	84%
	Crvena	75	25	75%
	Tišina	81	19	81%
5	Plava	76	24	76%
	Crvena	66	34	66%
	Tišina	77	23	77%
6	Plava	82	15	82%
	Crvena	71	29	71%
	Tišina	73	27	73%

Analizom prikupljenih podataka uočava se postojanje razlike prepoznavanja ključnih riječi ovisno o indivualnim osobama, a faktori koji utječu na raspoznavanje su boja glasa, naglasak, zvukovi iz okoline.



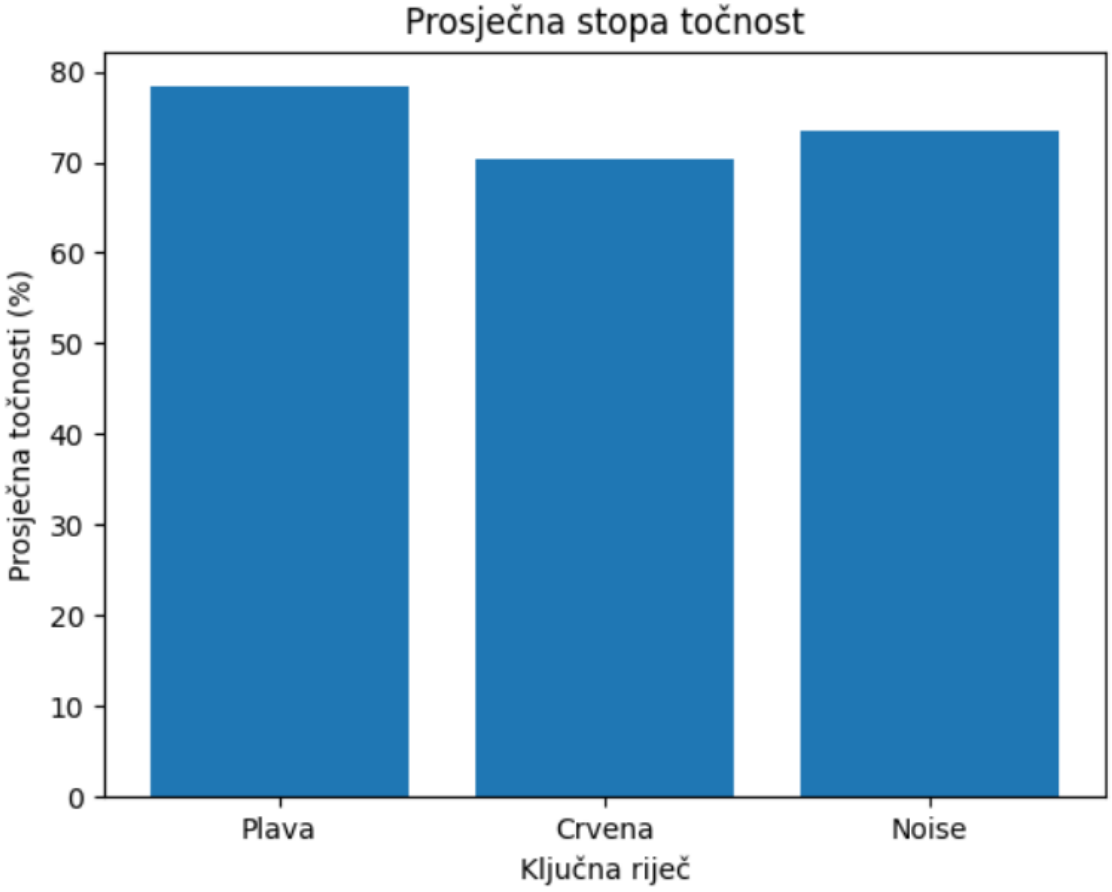
Sl. 5.1.Prikaz točnosti prepoznavanja ključnih riječi po osobi

Na prikazanoj slici *Sl. 5.1.* vidimo grafički prikaz podataka koje je mikroupravljački sustav, na koji je implementiran zadani model za raspoznavanje ključnih riječi „crvena“ i „plava“ te „tišina“, prepoznao.

Podatci u tablici *5.2* prikazuju prosječnu točnost prepoznavanja koja se bazira ta tablici 5.1. Grafički prikaz *Sl. 5.2.* pobliže prikazuje rezultate i daje bolji pogled na uspješnost stvaranja modela na vizualan način.

Tablica 5.2 Prosjek

Ključna riječ	Prosječna točnost (%)
Plava	78.3
Crvena	70.3
Tišina	73.5



Sl. 5.2. Prikaz prosječne stope točnosti ključnih riječi

Zaključak analize pokazuje da je sustav za prepoznavanje ključnih riječi postigao zadovoljavajuću razinu točnosti. Dodatna podešavanja modela ili poboljšanje kvalitete ulaznih podataka mogla bi povećati ukupnu točnost i pouzdanost sustava.

6. ZAKLJUČAK

U zaključku ovog rada, istražena je i prikazana primjena strojnog učenja na mikroupravljačima, što je postignuto korištenjem Arduino Nano BLE Sense platforme i Edge Impulse platforme za analizu audio podataka. Kroz implementaciju sustava koji prepoznaje ključne riječi poput "crvena" ili "plava" izgovorene od strane korisnika, demonstrirana je sposobnost mikroupravljača da se koristi za analizu zvuka i prepoznavanje govora. Korištenjem Arduino Nano BLE Sense pločice omogućena je integracija senzora i obrada audio signala na samoj mikroupravljačkoj platformi. Edge Impulse platforma pružila je alate za prikupljanje, analizu i obradu audio podataka te treniranje modela strojnog učenja za prepoznavanje ključnih riječi. Implementacija sustava za prepoznavanje ključnih riječi predstavlja korak prema razvoju inteligentnih ugrađenih sustava koji mogu interaktivno komunicirati s korisnicima putem glasovnih naredbi. Ova tehnologija može imati široku primjenu u područjima poput pametnih domova, nosive tehnologije, medicinskih uređaja i automatizacije industrijskih procesa.

Zaključno, ovaj rad ilustrira mogućnosti strojnog učenja na mikroupravljačima kroz konkretnu primjenu u analizi zvuka i prepoznavanju govora. Integracija tehnologija poput Arduino Nano BLE Sense pločice i Edge Impulse platforme omogućuje razvoj inovativnih aplikacija koje unapređuju interakciju između ljudi i tehnologije.

LITERATURA

[1] „Machine learning, explained | MIT Sloan“ [online], 15-svi-2024. Dostupno na: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained. [Pristupljeno: 17.5.2024.].

[2] Banerjee, R, *Hands-on TinyML Harness the power of Machine Learning on the edge devices*, sv. First. BPB Online: London, 2023.

[3] „mikroupravljač - Tehnički leksikon“ [online]. Dostupno na: https://tehnicki.lzmk.hr/clanak/mikroupravljac. [Pristupljeno: 16.8.2024.].

[4] „Arduino Nano 33 BLE“ [online]. Dostupno na: https://store.arduino.cc/products/arduino-nano-33-ble. [Pristupljeno: 1.3.2024.].

[5] „For beginners | Edge Impulse Documentation“ [online], 04-tra-2024. Dostupno na: https://edge-impulse.gitbook.io/docs/readme/for-beginners. [Pristupljeno: 17.5.2024.].

[6] „Learning on the edge“ [online], 04-lis-2022. Dostupno na: https://news.mit.edu/2022/machine-learning-edge-microcontroller-1004. [Pristupljeno: 16.8.2024.].

[7] „1.6. Nearest Neighbors“ [online]. Dostupno na: https://scikit-learn/stable/modules/neighbors.html. [Pristupljeno: 17.5.2024.].

SAŽETAK

Naslov: *Primjena strojnog učenja na mikroupravljačkim sustavima*

Ovaj rad predstavio je jedan od načina implementacije modela strojnog učenja na mikroupravljač razvojne pločice Arduino Sense 33 BLE. Model koji je implementiran napravljen je da prepoznaje ključne riječi „crvena“ i „plava“, te da prepoznaje ostale zvukove i govor koje su kategoriziranje pod „tišina“. Kada su ključne riječi prepoznate na pločici se pali crveno ili plavo LED svjetlo. Kroz ovaj rad prikazuje se proces stvaranja modela za audio raspoznavanje koji je kreiran u platformi Edge impulse. Platforma je jednostavna i kompaktna za korištenje kao što je prikazano u radu. Kroz rad predstavljena je Edge impulse platforma te su obuhvaćene njene funkcionalnosti potrebne za stvaranje modela za audio prepoznavanje. Analiziran je model i njegova efikasnost i predstavljena je razlika između modela prije stavljanja na arduino mikroupravljač i poslije kako bi se prikazala razlika modela prije nego što je konvertiran u oblik koji se može koristiti.

Ključne riječ: *Arduino IDE, Edge Impulse,LED, mikroupravljač, strojno učenje*

ABSTRACT

Title: *Application of Machine Learning on Microcontroller Systems*

This paper presented one of the methods of implementing a machine learning model on the Arduino Sense 33 BLE development board. The implemented model was designed to recognize the keywords "red" and "blue", as well as to categorize other sounds and speech under "silence". When these keywords are recognized, a red or blue LED light is illuminated on the board. This paper demonstrates the process of creating an audio keyword recognition model using the Edge Impulse platform, which is shown to be user-friendly. The model and its effectiveness were analyzed, and the difference between the model before deployment on the Arduino microcontroller and after is presented to illustrate the transformation of the model into a deployable format.

Keywords: *Arduino IDE, Edge Impulse,LED. machine learning, microcontroller,*

ŽIVOTOPIS

Helena Katalinić rođena 23.10.2000. u Požegi. Nakon završetka osnovnoškolskog obrazovanja u osnovnoj školi Antun Kanižlić Požega upisuje opću gimnaziju naziva Gimnazija Požega. Nakon četverogodišnjeg obazovanja upisuje smjer preddiplomski sveučilišni studij Elektrotehnika i Informacijska tehnologija.

_____

Potpis autora