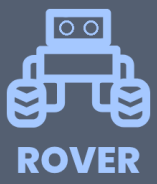


ROVER

Tehnička dokumentacija

Autori:

Luka Turkeš

Gabriel Zrilić

Mentorica:

Matea Filipović

Sadržaj

Uvod

O autorima

Luka Turkeš

Zovem se Luka Turkeš, i trenutno sam učenik četvrtog razreda Strukovne Škole Vice Vlatkovića, smjer Tehničar za računalstvo. Tijekom mog školovanja, stekao sam dosta znanja, ne samo putem formalnog obrazovanja, već i kroz samostalno učenje. Posebno sam se usavršavao u poljima digitalne elektronike i programiranja, uključujući se u razne projekte. Jedan od najistaknutijih projekata je moj 8-bitni računalni sustav, koji je u potpunosti izgrađen od integriranih krugova na breadboardovima. Također, sudjelovao sam u brojnim online natjecanjima u programiranju.

Gabriel Zrilić

Ja sam Gabriel Zrilić, učenik četvrtog razreda Strukovne škole Vice Vlatkovića u Zadru, smjer Tehničar za računalstvo. Digitalnu elektroniku sam učio u školi, ali sam također istraživao od kuće. Programiranju sam poprilično posvetio pažnju. Volim rješavati probleme i istraživati kako ih što optimiziranije riješiti. Osim toga samostalno radim na mnogim projektima u različitim tehnologijama. Sudjelovao sam u online natjecanjima iz programiranja i Infokup natjecanja iz različitih kategorija te postigao odlične rezultate.

Ideja projekta

Tijekom četiri godine strukovnog obrazovanja uočili smo jedan problem. Iako pohađamo strukovni smjer, naša škola ne nudi radnu praksu ili praktične radove. Stoga smo pronašli rješenje i razvili projekt koji će potaknuti učenike na kreativnost te im pokazati važnost open-source projekata.

Projekt ROVER (Remote Open-source Vehicle for Educational Research) ne samo da demonstrira naše znanje, već zbog svoje skalabilnosti može poslužiti kao platforma za učenje programiranja, elektronike i 3D modeliranja. Korisnici ove platforme mogu se osloniti na našu dokumentaciju, ali će isto tako biti potaknuti na samostalno istraživanje putem interneta.

Projekt je namijenjen svima koji žele razvijati gore navedene vještine. Osim toga, korisnici će imati priliku učiti o timskom radu. S obzirom na to da je nerealno očekivati da mladi već imaju sve potrebne vještine, formirat će se grupe koje će zajedno raditi na projektu, svatko unutar svog područja interesa.

Kako će se učenje odvijati? ROVER je open-source vozilo kojim se upravlja putem aplikacije. Program je vrlo skalabilan, što olakšava čitanje koda i njegovo prilagođavanje. Za izradu modula potrebno je dizajnirati PCB, programirati kod za mikroupravljač, razviti dodatak za aplikaciju i izraditi kućište za modul. Iako se čini kao puno posla, za dvije do tri osobe to je sasvim izvedivo.

Problemi koje ROVER rješava

- Primjena teorijskog znanja
- Stvarno iskustvo s hardverom i softverom
- Razvoj kritičkog razmišljanja i rješavanja problema
- Kolaborativno učenje i timski rad
- Motivacija kroz igru i istraživanje

Mogućnosti

ROVER je vozilo koje se može upravljati mobilnom aplikacijom. Aplikacijom se upravlja kretanje i moduli. Moduli su *kockice*, koje se slažu na ROVER i time vozilo dobiva dodatnu funkcionalnost. Trenutno ROVER ima module mjerenja udaljenosti, matrix ekran, i demo modul. Ovi moduli osim što dodaju funkcionalnost, pokazuju mogućnosti razvoja modula.

Demo modul pokazuje osnovne mogućnosti ROVERA, a to je kontroliranje sklopovlja. Putem aplikacije se mogu kontrolirati tri LED diode, jedna RGB dioda i zujalica. Svaka komponenta se kontrolira aplikacijom.

Modul mjerenja udaljenosti koristi ultrasonični modul koji pokazuje udaljenost na mobilnoj aplikaciji. Ovaj modul pokazuje ulazno/izlazne mogućnosti ROVERA, odnosno osim što ROVER može slati podatke modulu, može ih i primati.

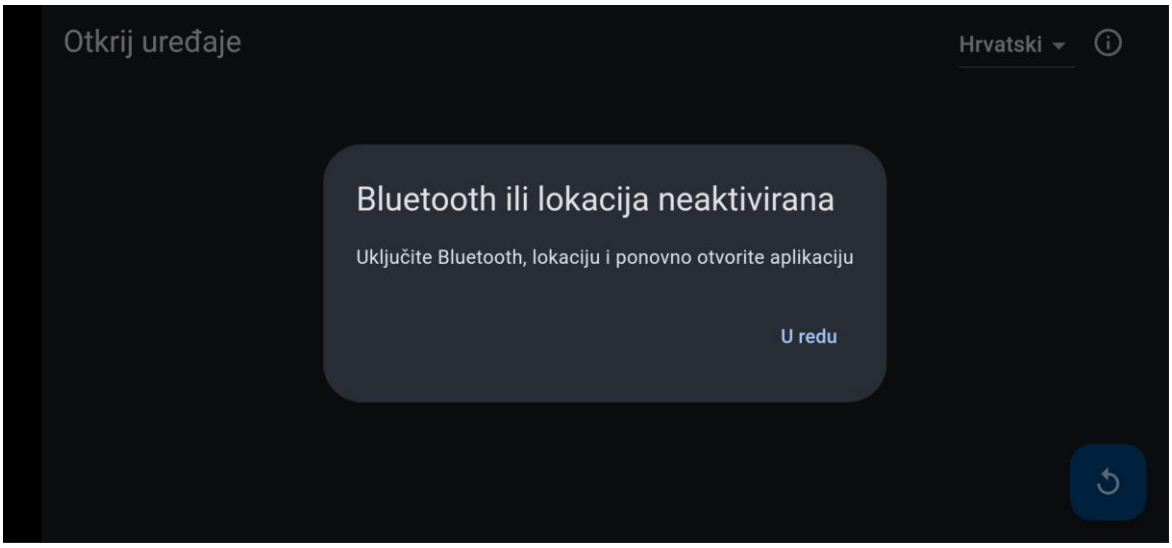
Zadnji modul napravljen je matrix modul. On može prikazati sliku nacrtanu preko mobilne aplikacije. Ovo pokazuje mogućnost serijskog prijenosa podataka na module.

Na dokumentaciju se često ne ulaže dovoljno vremena, ali ona je u nekim slučajevima i jedini izvor informacija. Zato smo odlučili odabrati tehnologije koje su veoma dobro dokumentirane i odlučili smo sami napraviti kvalitetan uvod u stvaranje modula kako bi korisnicima bilo što lakše se uključiti u ovaj projekt.

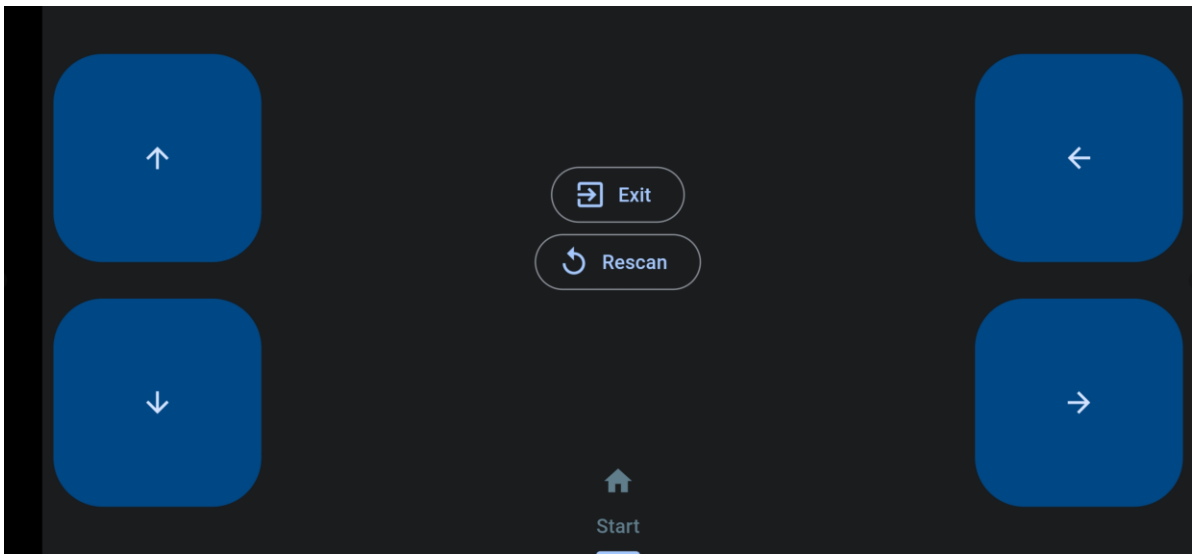
Detaljan način korištenja

Paljenje i spajanje aplikacije i vozila

Prije pokretanja aplikacije, uključite Bluetooth i lokacijske usluge na vašem uređaju kako bi aplikacija pravilno funkcionirala. Ukoliko neki od navedenih servisa nije aktiviran, pojavit će se upozorenje, a aplikacija će se ugasiti.



Nakon što omogućite pristup Bluetooth-u i lokaciji, aplikacija će započeti pretragu za ROVER-om. Pritiskom na gumb pokrećemo skeniranje uređaja, što omogućuje prikaz svih dostupnih uređaja s kojima se možemo povezati. Odaberite ROVER i nakon uspješnog spajanja, ekran će se prebaciti na upravljačko sučelje.

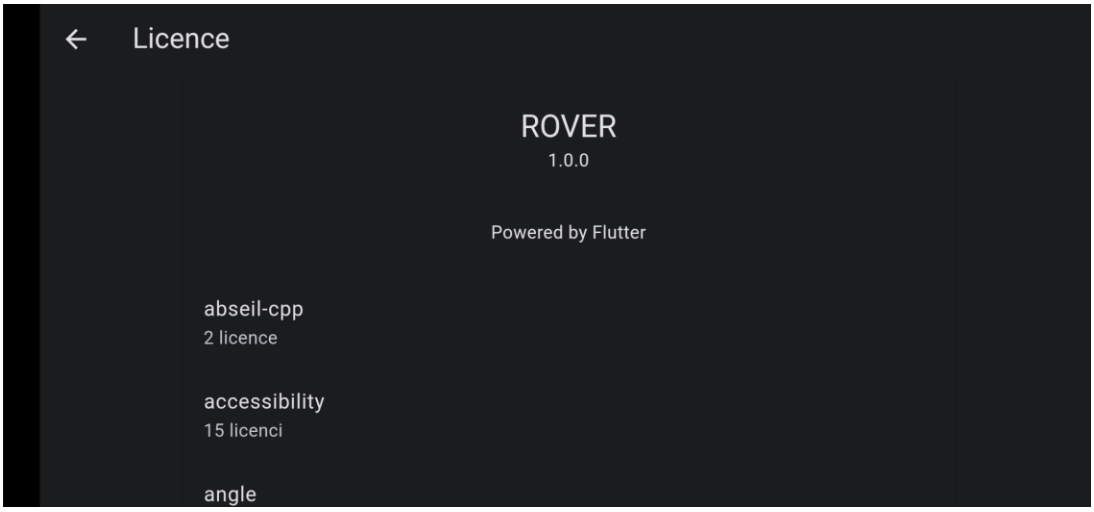


Mijenjanje jezika te informacije o aplikaciji

Pritiskom na meni možete odabrati jezik sučelja. Trenutno je aplikacija prevedena na hrvatski i engleski, no zbog svoje jednostavne arhitekture, omogućeno je dodavanje još jezika.

Pritiskom na gumb u gornjem desnom kutu (info) prikazat će se informacije o aplikaciji, uključujući verziju, linkove do GitHub repozitorija, web verzije dokumentacije, pravila o privatnosti te licence. U licencama su navedeni svi paketi i programi koji su korišteni pri stvaranju aplikacije.

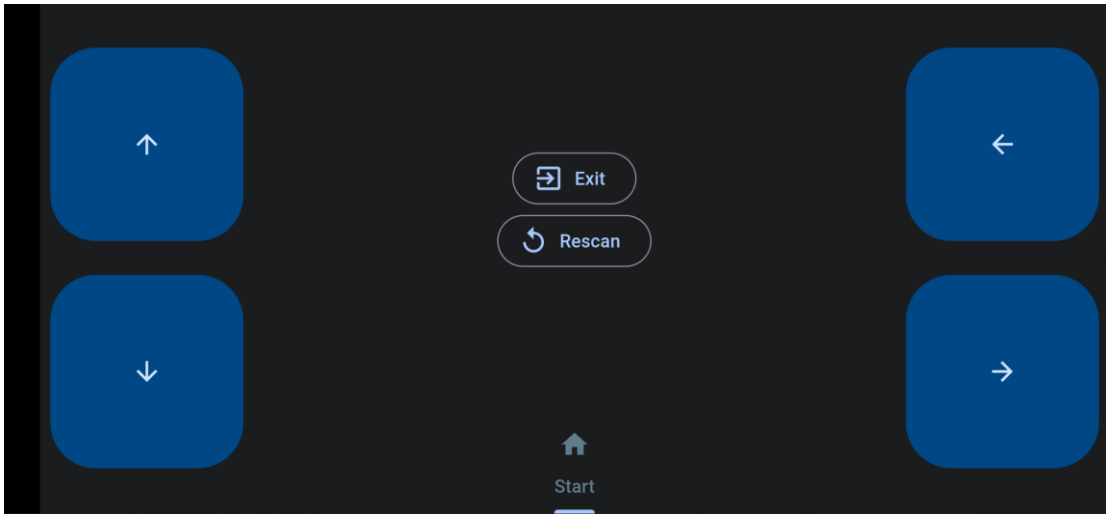
U licencama su navedeni svi paketi i programi koji su korišteni pri stvaranju aplikacije.



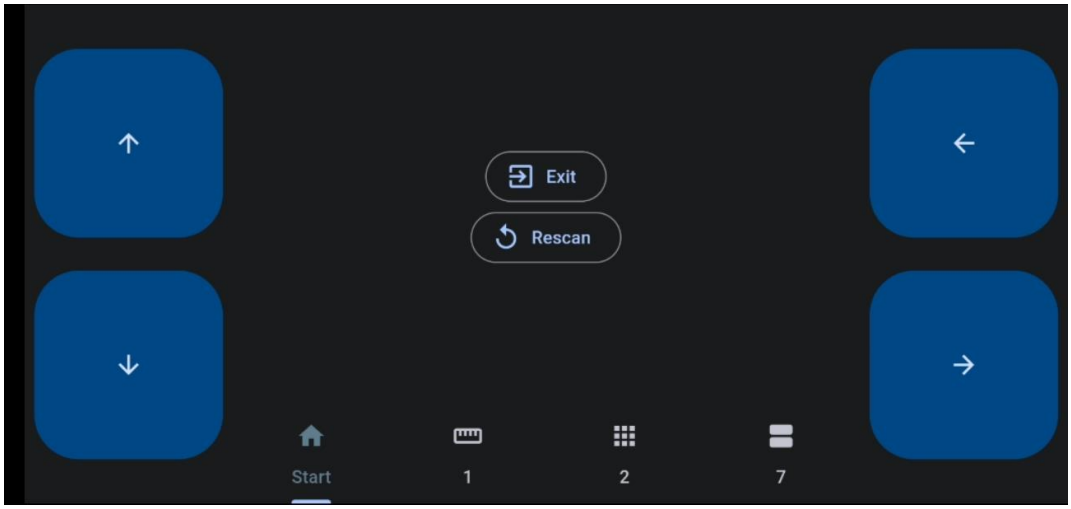
Upravljanje ROVER-om te skeniranje

Na slici je prikazano upravljačko sučelje za ROVER. Sa lijeve i desne strane su postavljeni gumbi () koji služe za pokretanje ROVER-a. U sredini se nalaze upravljačke ploče modula (u daljnjem tekstu nazvane panelima). Uvijek postoji osnovni panel koji služi za skeniranje ROVER modula. Kada se na tom panelu pritisne gumb „Skeniraj“ (Rescan), ROVER ponovno skenira i prikaže sve spojene module. Spojene module prikazuje u novim karticama, a kada se odabere neka od njih pritiskom na njihovu ikonu, panel se promjeni za taj određeni modul.

Ovako izgleda ekran kada nema spojenih modula:



Ovako izgleda ekran s 3 spojena modula:

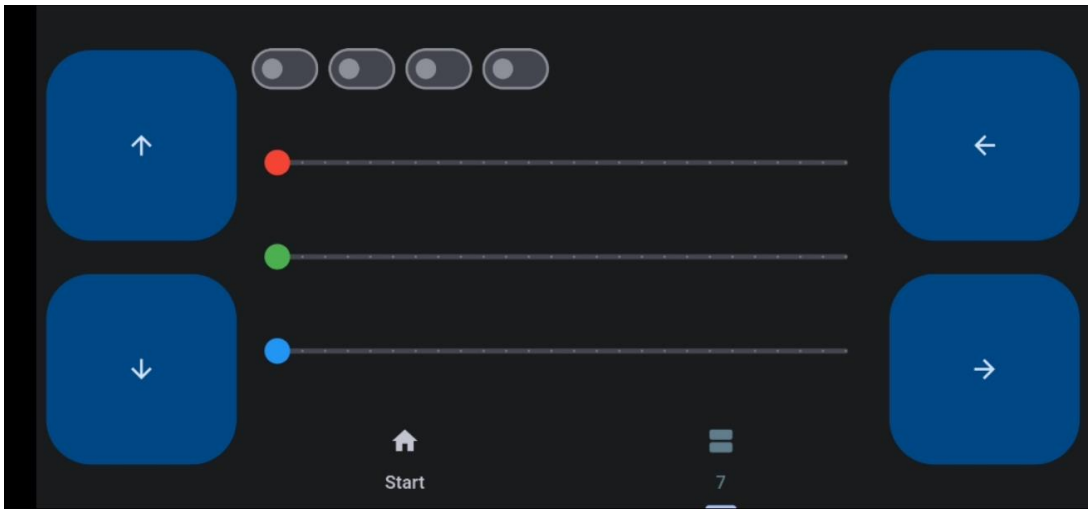


Moduli

Trenutno postoje tri modula, ali ih se planira napraviti još. Ti moduli su: “Demo modul”, “Ultrasonični modul” i “Matrix modul”.

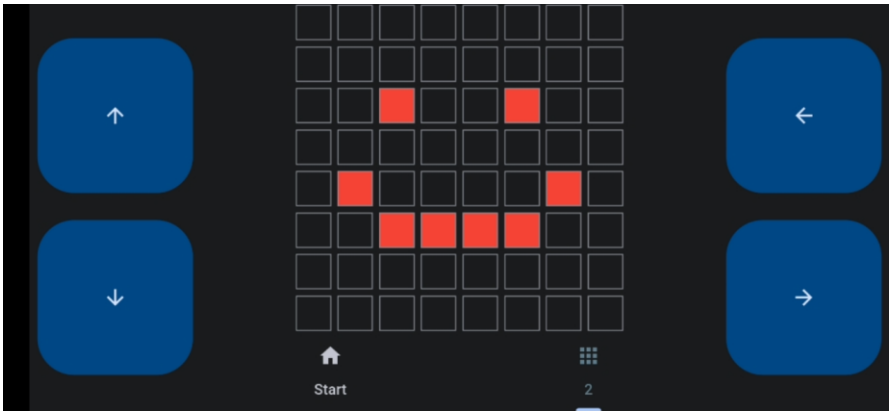
Demo modul

Demo modul sadrži diode i zvučnicu koje se kontroliraju. Na sučelju aplikacije postoje kontrole za svaku navedenu komponentu. Prekidači upravljaju jednobojnim diodama, dok klizači upravljaju bojom RGB diode. Gumb upravlja zujalicom. Pritiskom na te određene komponente grafičkog sučelja upravljate Demo modulom.



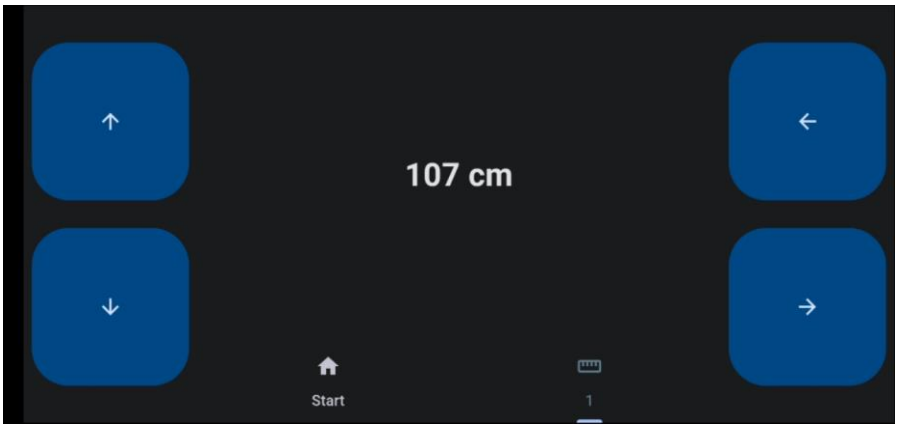
Matrix modul

Sadrži matrix ekran prikazan u grafičkom sučelju kao 8x8 tablica (1). Pritiskom na određeno polje slika na mobilnoj aplikaciji se iscrtava na ROVER modulu. Kada se prebacite na drugi modul, stanje Matrix modula ostaje nepromijenjeno.



Ultrasonični modul

Prikazuje udaljenost od neke prepreke u centimetrima.



Izlazak iz aplikacije

Pritiskom na gumb se prikazuje upit za potvrdu za izlazak iz aplikacije.

Tehnička dokumentacija

Softver

Lista značajki aplikacije

Početni ekran

- Skeniranje i spajanje s vozilom
- Prikaz informacija o aplikaciji
- Prijevod

Upravljanje

- Skeniranje spojenih modula
- Upravljanje kretanja vozila
- Upravljanje različitih modula

Moduli

Demo modul

- Upravljanje 3 jednobojne LED diode
- Upravljanje RGB diode
- Upravljanje zujalice

Matrix modul

- Upravljanje matrix ekrana aplikacijom i serijskim prijenosom
- Crtanje oblika na matrici

Ultrasonični modul

- Ulazno izlazna mogućnost podataka
- Prikaz udaljenosti

Sistemska konfiguracija

Neće svi uređaji biti u stanju upotrebljavati ovu aplikaciju. Navedeno je više sistemskih konfiguracija, jedna za mobitele te je jedna za računala. Konfiguracija za računala je potrebna samo ako mislite stvarati module.

Preporučena sistemska konfiguracija za mobilne uređaje

- Mobilni uređaj ili tablet s operacijskim sustavom Android
- RAM memorija 4 GB
- 30 MB slobodnog prostora (1 GB slobodnog prostora za razvoj modula)
- Bluetooth Classic

Preporučena sistemska konfiguracija za računala

- Procesor s 4 jezgre i radnim taktom 3 GHz
- RAM memorija 8 GB
- 10 GB ili više slobodnog prostora
- Operacijski sustav Windows, Linux ili MacOS

Potreban softver

Windows

- Android Studio
- Flutter SDK
- Raspberry Pi Pico C/C++ SDK
- Visual Studio Code

Linux

- Android Studio
- Flutter SDK
- Raspberry Pi Pico C/C++ SDK
- Visual Studio Code
- CMake
- GNU GCC

MacOS

- Android Studio
- Flutter SDK
- Raspberry Pi Pico C/C++ SDK
- Visual Studio Code
- CMake
- GNU GCC

Tehnologije

Projekt ROVER napravljen je pomoću više tehnologija. Odabrane su tehnologije koje su popularne, često održavane i jednostavne za naučiti.

Za samu aplikaciju smo koristili Flutter. To je open-source SDK (software development kit) napravljen od strane Google-a. Koristi se za programiranje brzih cross-platform aplikacija. Unatoč tome ova aplikacija se koristi samo na android uređajima jer iOS nema mogućnost korištenja Bluetooth Classic. Za programiranje logike i UI-a se koristi Dart programski jezik.

Mikroupravljač koji smo koristili je bio Raspberry Pi Pico. On ima mogućnost programirati se u Micropythonu ili C/C++ pomoću SDK-a. Odabrali smo programirati u C-u jer je to programski jezik s kojim smo upoznati i u usporedbi s Micropythonom je puno brži.

GitHub koristimo za kontrolu verzije. Napravili smo organizaciju u kojoj smo stavili repozitorije projekta.

Optimizacija

Uložen je dodatan trud u optimizaciju ROVERA, i sa strane aplikacije i mikroupravljač. Putem bluetooth-a komunicira slanjem već dogovorenim formatom poruke. Aplikacija je pisana asinkrono, što ubrzava aplikaciju posebice u djelovima slanja i primanja poruka.

Instalacija aplikacije

Trenutno postoji dva načina za instaliranje aplikacije. Ako ne planirate nadograditi ROVER ili njegovu aplikaciju, preporučuje se preuzimanje APK datoteke s GitHub repozitorija. No, ako želite modificirati aplikaciju ili ROVER, preporučuje se kloniranje repozitorija na računalo i otvaranje projekta u razvojnom okruženju poput VSCode-a ili Android Studija.

Kada je u pitanju učitavanje koda na mikroupravljač, postupak je sljedeći: klonirate repozitorij i prenesete main.uf2 datoteku na mikroupravljač. Kod se prenosi tako da tijekom spajanja Raspberry Pi Pico putem USB-a držite BOOTSEL gumb. Na računalu će se pojaviti nova jedinica za pohranu na koju ćete kopirati main.uf2 datoteku.

Sigurnost

Sigurnost ove aplikacije je potpuno zajamčena jer ne koristi nikakve online servise. Također, i pristupni podaci i algoritmi korišteni za komunikaciju su sigurni, a provjerava se i točnost formata. U slučaju netočnih formata, neće se dogoditi nikakav neželjeni ishod. Dodatno, budući da je cijeli kod otvorenog koda (open-source), korisnici imaju potpunu transparentnost i mogu proučiti svaki dio prije instalacije.

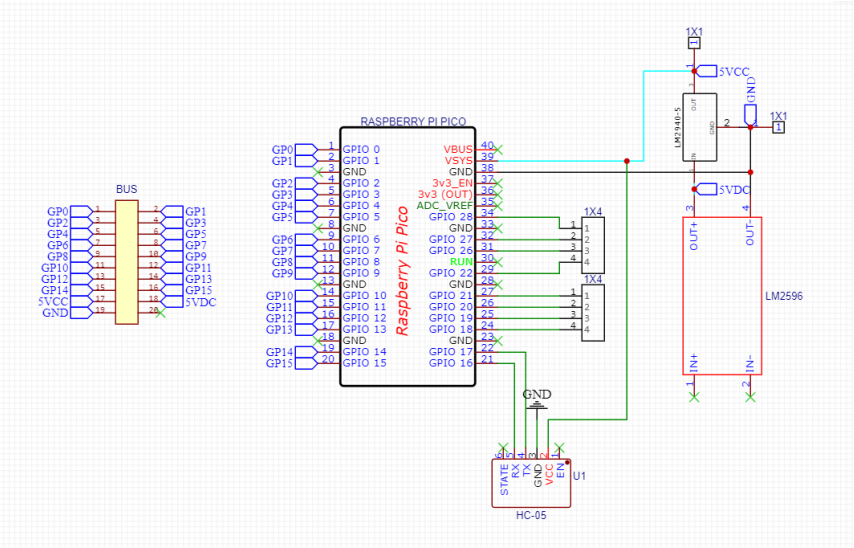
Hardver

Popis dijelova i tehnologija

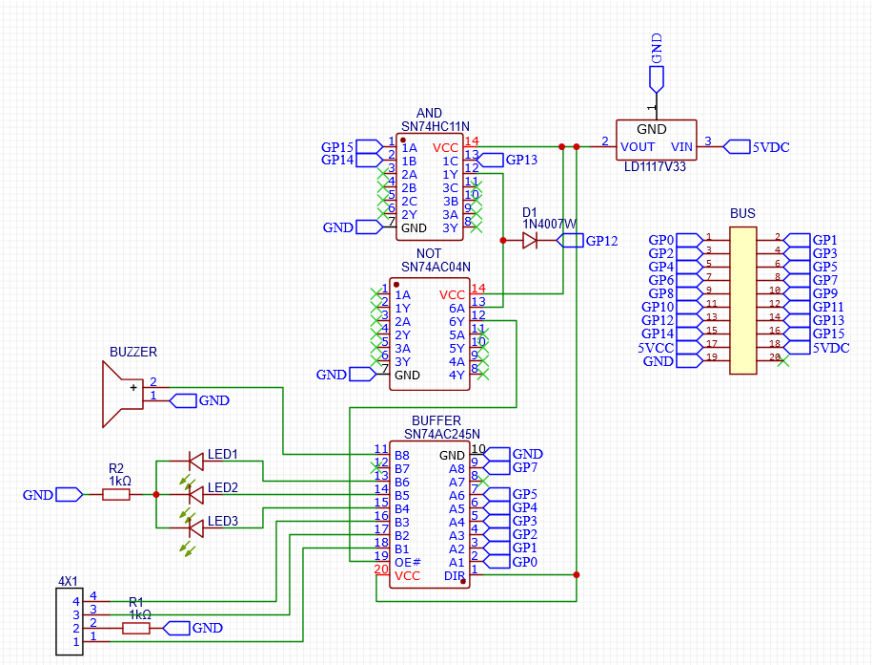
- Mikroupravljač - Raspberry Pi Pico, koji omogućuje programiranje u C/C++
- Motor - JGA25-370 motor na 12V 100RPM x4
- Upravljač motora BTS7960B x2
- Komunikacijski modul – HC-05 (Bluetooth Classic)
- Komponente - SN74AC245N x2, SN74AC244N x1, SN74AC04N x3, SN74HC11N x3, LD1117V33 x3, 1kΩ resistor x2, LED diode x3, header pins, LM2596 buck-converter x1, LM2940-5 x1
- Senzori i dodatni djelovi - 8x8 Matrix, Buzzer, Ultrasonic sensor
- Napajanje 12V baterija
- 3D modeli

Elektroničke sheme

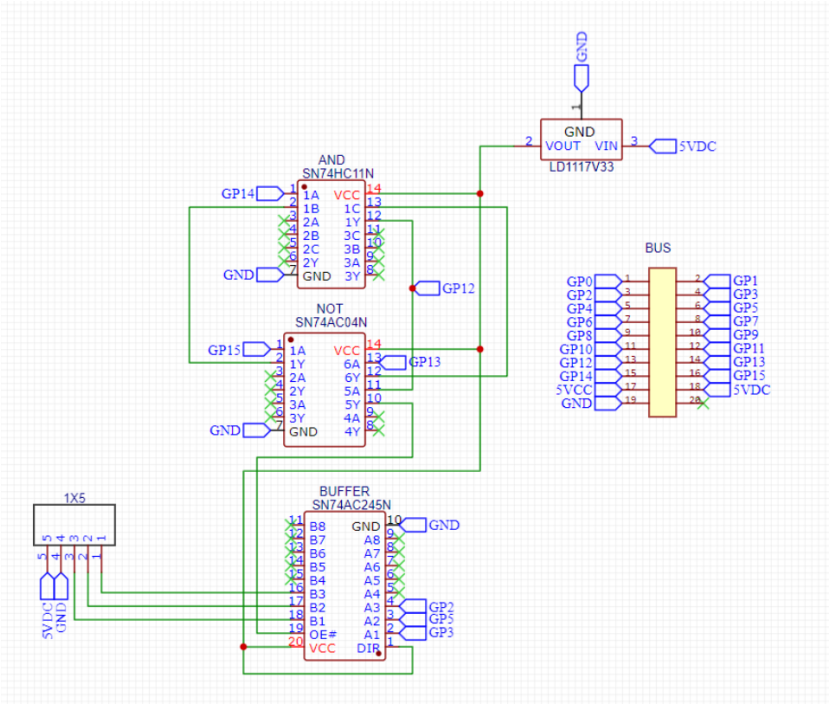
Glavna shema



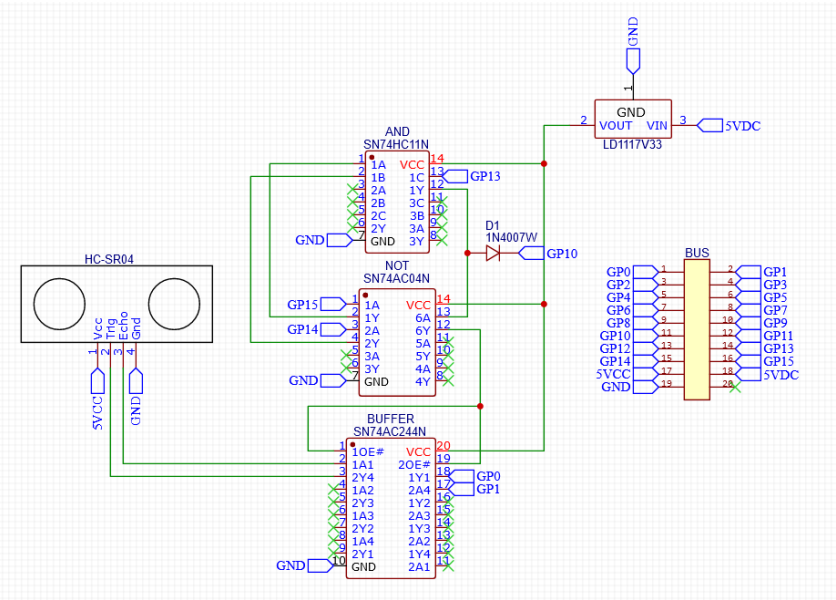
Demo modul



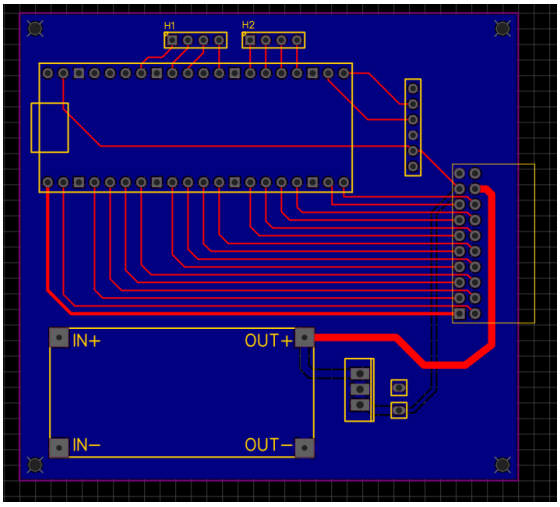
Matrix modul



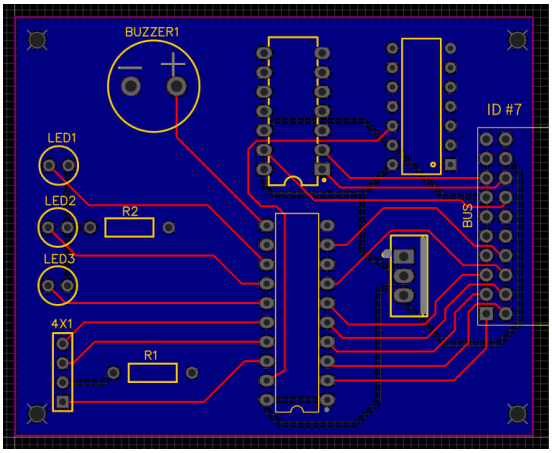
Ultrasonični modul



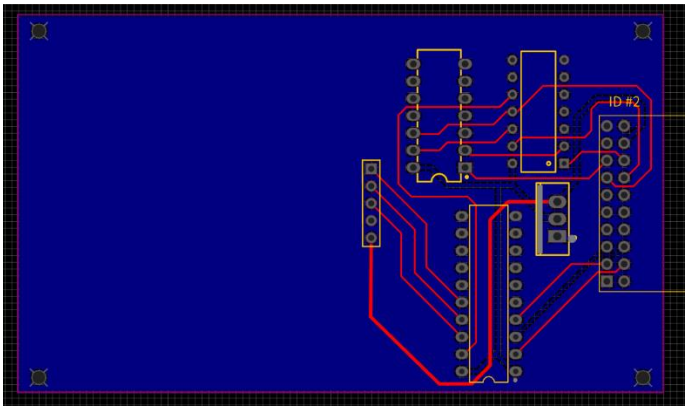
PCB



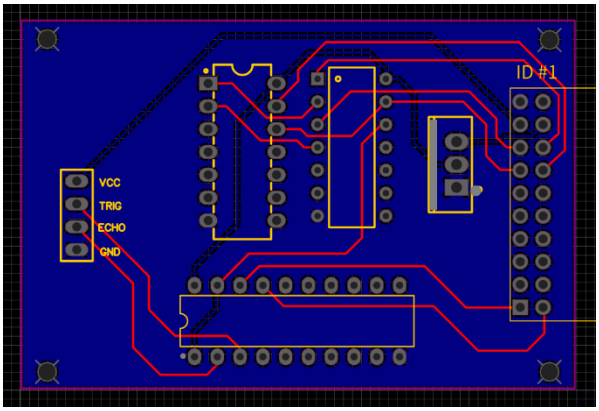
Glavni PCB



Demo modul

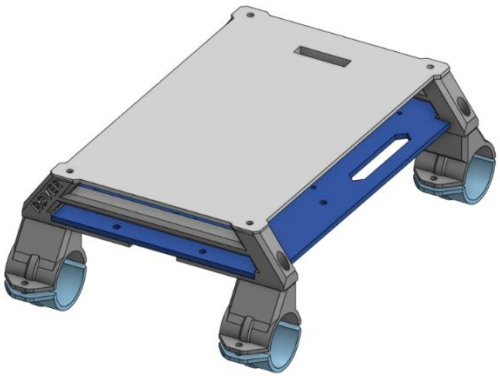


Matrix modul

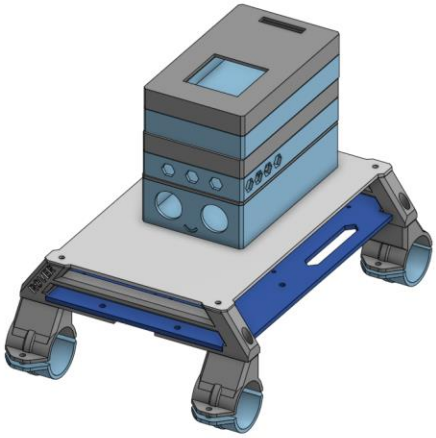


Ultrasončni modul

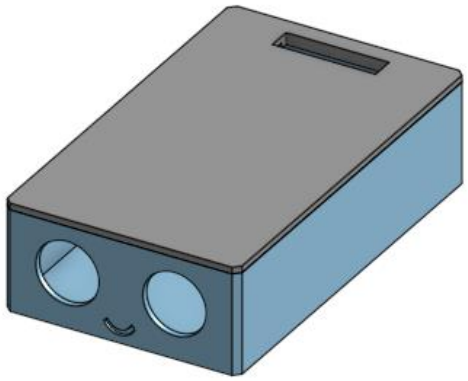
3D modeli



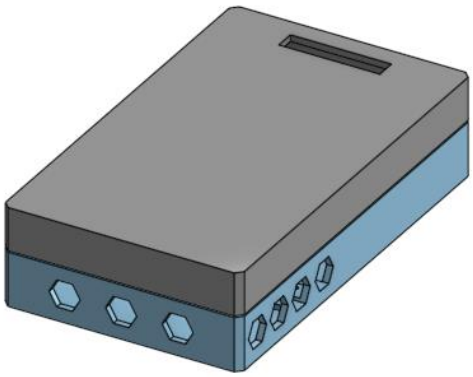
ROVER bez modula



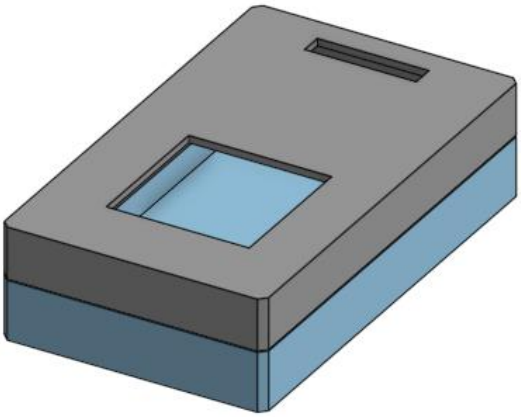
ROVER sa svim modulima



Ultrasonični modul



Demo modul



Matrix modul

Proces izrade

Prva faza: izrada mobilne aplikacije

U prvoj fazi razvoja mobilne aplikacije, odlučili smo koristiti Flutter kao razvojno okruženje. Budući da Flutter ne podržava Bluetooth funkcionalnost, bili smo primorani koristiti programski paket flutter_blue_serial. Iako je to bila jedina opcija za integraciju Bluetooth Classic tehnologije, naišli smo na izazov zbog nedostatka dokumentacije. Morali smo temeljiti naše razumijevanje paketa na primjerima i vlastitom istraživanju kako bismo shvatili njegovo funkcioniranje.

Kako bismo olakšali razvoj ne samo nama već i budućim korisnicima, napisali smo funkcije za upravljanje Bluetooth vezom. Ovo je omogućilo jednostavnije implementiranje Bluetooth komunikacije u našu aplikaciju.

Za testiranje funkcionalnosti aplikacije, izradili smo malu platformu za "mock-testing". Povezali smo Raspberry Pi Pico mikroupravljač s HC-05 Bluetooth čipom i programirali ga da generira unaprijed definirane poruke i podatke. Ovo testiranje omogućilo nam je provjeru ispravnosti Bluetooth komunikacije u kontroliranom okruženju prije nego što smo se upustili u testiranje s pravim uređajima.

Druga faza: izrada upravljačkog koda na Raspberry Pi Pico-u

U drugoj fazi razvoja, nakon što smo utvrdili da je većina funkcionalnosti aplikacije spremna za upotrebu, usredotočili smo se na pisanje upravljačkog koda za Raspberry Pi Pico. Budući da veći dio hardvera nije bio dostupan u tom trenutku, pristupili smo programiranju tako što smo kod podijelili u segmente koje smo zasebno testirali.

Ovaj pristup omogućio nam je temeljito testiranje svake komponente prije nego što smo je integrirali s ostatkom koda. Kada bismo utvrdili da određena komponenta pravilno funkcionira, izolirali bismo je od ostatka koda kako bismo osigurali njenu stabilnost i ispravnost u cjelokupnom sustavu.

Za testiranje izoliranih dijelova koda koristili smo aplikaciju koja je već bila dostupna. Ovo nam je omogućilo simuliranje stvarnog korisničkog iskustva i provjeru funkcionalnosti upravljačkog koda u kontroliranom okruženju prije nego što smo ga integrirali s ostatkom aplikacije.

Treća faza: izrada elektroničkih shema i dizajn PCB-a

Treća faza projekta uključuje izradu elektroničkih shema i dizajn tiskanih pločica (PCB-a) koristeći platformu EasyEDA. Početak ove faze obuhvaća detaljno planiranje i skiciranje elektroničke sheme koja predstavlja logičku strukturu i povezanost svih elektroničkih komponenti u projektu.

Nakon završetka sheme, proces se nastavlja pretvaranjem shematskog prikaza u fizički dizajn PCB-a unutar iste platforme. Omogućena je automatska konverzija shematskog dijagrama u nacrt PCB-a, nakon kojeg je potrebno ručno prilagođavanje rasporeda komponenti, tragova, slojeva i drugih elemenata kako bi ostvarili traženu funkcionalnost.

Završni korak u ovoj fazi uključuje pregled i finalizaciju dizajna PCB-a, pri čemu se provjeravaju sve dimenzije, povezanosti i specifikacije komponenti kako bi se osiguralo da je sve u skladu s tehničkim zahtjevima i standardima. Nakon zadovoljavajuće provjere, dizajn se izvozi u standardnom formatu za proizvodnju, poput Gerber datoteka, koje se zatim mogu koristiti za naručivanje gotovih PCB-a od proizvođača.

Četvrta faza: 3D modeliranje

Četvrta faza projekta obuhvaća 3D modeliranje koristeći Onshape, potpuno cloud-bazirana CAD platforma koja omogućava korisnicima izradu preciznih 3D modela komponenata i sklopova unutar samog preglednika. Proces započinje definiranjem i skiciranjem osnovnih oblika i dimenzija svih potrebnih dijelova, uključujući glavno tijelo ROVER-a, nosače, i druge strukturne elemente, a završava izvozom modela u STL formatu, koji je kompatibilan s 3D printerima.

Završni proizvod

Kada imamo sve dijelove i potreban kod, bilo je potrebno samo složiti. Ovo je sada spremo za uporabu i daljnje modificiranje.

U potpunosti sastavljan ROVER sa svim spojenim modulima

ROVER bez modula

Sučelje za komunikaciju modula

Matrix modul

Demo modul

Ultrasonični modul

Sastavljanje

Sastavljanje je veoma jednostavno, nema puno vijaka i ljepila se koristi jako malo. Zbog svoga načina izrade je veoma modularan.

Stvaranje modula

Osnove komuniciranja

U komuniciranju se koristi bluetooth classic. Za komunikaciju između HC-05 i mikroupravljača koristi se UART protokol. U oba smjera se šalje niz podataka s dogovorenim mjestima.

Podaci koji se šalju s aplikacije mikroupravljaču:

254 označava početak poruke, ID označava koji modul se koristi ili koja glavna akcija se događa. Podatak motori sadrži broj u kojem pozicija bita označava je li pojedinačan gumb smjera pritisnut ili ne. Ostalo su podaci za modul.

Podaci koji se šalju s mikroupravljača prema aplikaciji:

ID označava koji modul šalje podatak, a sve ostalo su podaci koje modul šalje.

Funkcije za komunikaciju (Flutter)

void messageReaction(List<int> message);

Funkcija koja je potrebna modificirati se ako modul ima povratnu poruku (npr. Ultrasonični modul)

```
/// Pridruži funkcije zadanom ID-u i modulu
///
/// Na osnovu prvog polja biti će drugačija reakcija
/// Reakcija se dodaje samo modulima koji imaju povratnu poruku roveru
void messageReaction(List<int> message) {
  switch (message[0]) {
    case 1:
      /// Ultrasonic modul
      ultrasonicModuleProvider.getDistance(message);
      break;
    case 2:
      /// Matrix modul
      /// Nema povratnu informaciju
      break;
    case 3:
      /// ID je neiskorišten
      break;
    case 4:
      /// ID je neiskorišten
      break;
    case 5:
      /// ID je neiskorišten
      break;
    case 6:
      /// ID je neiskorišten
      break;
    case 7:
      /// Demo modul
      /// Nema povratnu informaciju
      break;
    case 17:
      /// Ponovno skeniranje
      _panelsProvider.updateLists(message);
      break;
    default:
      dev.log('no case');
      break;
  }
  notifyListeners();
}
```

Ako želite da vaš željeni modul daje podatke i aplikacija reagira, stavite dodatak koda u određeni slučaj.

changeDataForModule(List<int> list);

Mijenjanje podataka koja se šalje za modul. Podaci su izraženi u listi.

```
/// Promjeni [dataForModule]
///
/// Promjeni samo dio koji se ondosi na podatke za modul
void changeDataForModule(List<int> list) {
  for (int i = 0; i < list.length; i++) {
    dataForModule[i] = list[i];
  }
}
```

sendMessage({bool changingModule = false});

Šalje podatke koji se nalaze u dataForModule. Sadrži opcionalni argument changingModule, koji je na početku netočan.

```
/// Pošalji poruku roveru
void sendMessage({bool changingModule = false}) async {
  sending = true;
  Uint8List message;
  if (!changingModule) {
    message = Uint8List.fromList([254, mode, motorControl] + dataForModule);
  } else {
    message = Uint8List.fromList(
        [254, 19, motorControl, mode] + List.filled(61, 0));
  }

  dev.log('$message');

  try {
    connection!.output.add(message);
    await connection!.output.allSent;
  } catch (e) {
    dev.log('Catch in: void sendMessage()');
  }
  sending = false;
}
```

Funkcije za komunikaciju (Raspberry Pi Pico)

void response():

Potrebno je modificirati, to jest, dodati funkcije koje će se izvršiti kao odgovor za poruku s određenim primljenim ID-em.

```
void response() {
    gpio_put(25, uart_data_waiting);
    motor_driver(input_buffer[1]);
    switch (input_buffer[0]) {
    case 0:
        // neutral
        break;
    case 1:
        // ultrasonic_module
        ultrasonic_module_reaction();
        bluetooth_send();
        break;
    case 2:
        // matrix_module
        matrix_module_reaction();
        break;
    case 3:
        // ID nije korišten
        break;
    case 4:
        // ID nije korišten
        break;
    case 5:
        // ID nije korišten
        break;
    case 6:
        // ID nije korišten
        break;
    case 7:
        // demo_module
        demo_module_reaction();
        break;
    case 18:
        // rescan
        scan_for_modules();
        send_return_message();
        break;
    case 19:
        // Promjena modula
        set_module_id(input_buffer[2]);
        module_setup(input_buffer[2]);
        break;
    default:
        break;
    }
}
```

uint8_t get_input_buffer();*

Pristup svim podacima koji su poslani od strane aplikacije.

```
uint8_t* get_input_buffer() {
    return input_buffer;
}
```

uint8_t get_output_buffer();*

Pristup i postavljanje svih podataka podacima koji će se slati aplikaciji.

```
uint8_t* get_output_buffer() {
    return output_buffer;
}
```

void bluetooth_send();

Slanje podataka koji se nalaze u output_buffer.

```
// Vrati poruku vezana za rescan
void send_return_message() {
    output_buffer[0] = 17;
    uint8_t *connected_modules = get_connected_modules();
    output_buffer[1] = get_number_of_modules();

    for(int i = 0; i<output_buffer[1]; i++) {
        output_buffer[2 + i] = connected_modules[i];
    }
    bluetooth_send();
}
```

Stvaranje dodatka u Flutter aplikaciji

Sav kod koji je potreban se nalazi u lib mapi. UI koji napravite stavite u mapu lib/panel_screens/screens. Sva dokumentacija za stvaranje UI-a je na službenim stranicama Fluttera. Ovo je primjer jednostavnog UI-a u Flutteru.

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:rover_app/providers/modules/ultrasonic_module_provider.dart';
import 'package:rover_app/providers/panels.dart';

/// UI ultrasonic modula
/// Sadrži prikaz podataka senzora udaljenosti
/// Koristi [UltrasonicModuleProvider]
class UltrasonicModulePanel extends StatelessWidget {
  const UltrasonicModulePanel({super.key});

  @override
  Widget build(BuildContext context) {
    // Ova linija je bitna, ona označava koji je ovo ID
    Provider.of<Panels>(context, listen: false).changeToModule(context, 1);

    return Consumer<UltrasonicModuleProvider>(
      builder: (context, provider, child) {
        return Center(child: Text('${provider.distance} cm', style: const TextStyle(fontSize: 30,
fontWeight: FontWeight.bold),));
      },
    );
  }
}
```

Što se upravljačkog djela koda ona se kontrolira providerom. Provider je upravljač stanja u Flutteru i jednostavno se koristi. Te datoteke se stavljaju u mapu lib/providers/modules

Ovo bi bio primjer jednostavnog provider-a:

```dart
import 'package:flutter/material.dart';
import 'package:rover_app/providers/bt_controller.dart';

/// Demo modul provider
///
/// Sadrži jedan ultrasonic modul
/// Šalje i prima podatke
/// Šalje u formatu
///   [trig]
/// Prima u formatu
///   [prvi broj][drugi broj]
///   udaljenost = prvi broj + drugi broj
class UltrasonicModuleProvider extends ChangeNotifier {
  int distance = 0;

  void getDistance(List<int> inputBuffer) {
    distance = inputBuffer[1] + inputBuffer[2];
    notifyListeners();
  }

  /// Pokreće funkciju koje je aktivna u pozadini cijelo vrijeme
  /// Svake pola sekunde pošalji ping za osvježavanje udaljenosti ako je
  /// odabran taj modul
  void startUltrasonicService(BtController bt) async {
    while (true) {
      await Future.delayed(const Duration(milliseconds: 2500), () {
        if (bt.mode == 1) {
          bt.sendMessage();
        }
      });
    }
  }
}
```

Nakon što napišete ta dva djela, morate ih dodati u program. Zato dodajemo provider i UI u nekoliko dijelova koda:

main.dart

```
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);

  /// Provideri za module (dodajte ako mislite napraviti modul)

  DemoModuleProvider demoModuleProvider = DemoModuleProvider();
  UltrasonicModuleProvider ultrasonicModuleProvider =
      UltrasonicModuleProvider();
  MatrixModuleProvider matrixModuleProvider = MatrixModuleProvider();

  /// ---------------------

  /// Provideri za panele i Bluetooth
  Panels panels = Panels();
  BtController btController =
      BtController(panels, demoModuleProvider, ultrasonicModuleProvider, matrixModuleProvider);

  ///                                                 ^
  ///                                            Novi provider

  /// ...


  /// Paljenje aplikacije i postavljane je horizontalno
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.landscapeRight,
    DeviceOrientation.landscapeLeft,
  ]).then((value) {
    [
      Permission.location,
      Permission.bluetooth,
      Permission.bluetoothConnect,
      Permission.bluetoothScan
    ].request().then((status) {
      runApp(MultiProvider(providers: [
        ChangeNotifierProvider(create: (_) => btController),
        ChangeNotifierProvider(create: (_) => panels),
        ChangeNotifierProvider(create: (_) => demoModuleProvider),
        ChangeNotifierProvider(create: (_) => ultrasonicModuleProvider), /// <- Novi provider
        ChangeNotifierProvider(create: (_) => matrixModuleProvider)
      ], child: const MainApp()));
    });
```

panels.dart

```
  final Map<int, StatelessWidget> moduleById = {
    16: const MainPanel(),
    1: const UltrasonicModulePanel(),  /// <- Novi panel
    2: const MatrixModulePanel(),
    7: const DemoModulePanel()
  };
```

bt_controller.dart

```
  BtController(
    this._panelsProvider,
    this.demoModuleProvider,
    this.ultrasonicModuleProvider,  /// <- Novi provider
    this.matrixModuleProvider,
  ) {
    services();
```

Koristeći IDE nadopunite još mjesta što fali.

Stvaranje dodatka za mikroupravljač

Dodatak za mikroupravjač je lakše dodavati negu u Flutteru, kod se stavlja u src mapu, i doda linija u CMakeLists.txt

```
cmake_minimum_required(VERSION 3.13)

include(pico_sdk_import.cmake)
project(rover-pico C CXX ASM)

set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

pico_sdk_init()

# sve nove datoteke se dodaju u add_executable
add_executable(main
src/main.c
src/bluetooth.c
src/system_manager.c
src/demo_module.c
src/ultrasonic_module.c
src/matrix_module.c
)

pico_enable_stdio_usb(main 1)
pico_enable_stdio_uart(main 1)

target_link_libraries(main pico_stdlib hardware_pwm hardware_gpio hardware_spi hardware_timer
pico_multicore)

pico_add_extra_outputs(main)
```

Primjer stvaranja modula:

ultrasonic_module.c

```c
#include "ultrasonic_module.h"
#include "pico/multicore.h"

//component    gpio
//TRIG         0
//ECHO         1

void init_ultrasonic_module() {
    gpio_set_dir(TRIG, GPIO_OUT);
    gpio_set_dir(ECHO, GPIO_IN);
    gpio_set_dir(25, GPIO_OUT);
}

volatile int shared_distance = 0;

uint32_t pulseIn(uint gpio, uint level, uint32_t timeout) {
    // Označi početnio vrijeme
    absolute_time_t start_time = get_absolute_time();

    // Pričekaj da bude !level
    while (gpio_get(gpio) != level) {
        if (absolute_time_diff_us(start_time, get_absolute_time()) >= timeout) {
            return 0;
        }
    }

    // Označi početno vijeme pulsa
    absolute_time_t pulse_start_time = get_absolute_time();

    // Pričekaj da bude level
    while (gpio_get(gpio) == level) {
        if (absolute_time_diff_us(start_time, get_absolute_time()) >= timeout) {
            return 0;
        }
    }
    // Označi vrijeme kraja pulsa
    absolute_time_t pulse_end_time = get_absolute_time();

    // Izračunaj razliku vremena
    uint32_t pulse_duration = absolute_time_diff_us(pulse_start_time, pulse_end_time);

    return pulse_duration;
}

int get_distance() {
    // Započni puls
    gpio_put(TRIG, 0);
    sleep_us(2);
    gpio_put(TRIG, 1);
    sleep_us(10);
    gpio_put(TRIG, 0);

    // Izmjeri vrijeme
    uint32_t duration = pulseIn(ECHO, 1, 10000);

    // Izračunaj udaljenost
    int distance = (duration * 0.0343) / 2.0;

    return distance;
}
```

```
// Spremi udajenost u listu slanja
void ultrasonic_module_reaction() {
    uint8_t *message = get_input_buffer();
    get_output_buffer()[0] = 1;

    int distance = get_distance();

    get_output_buffer()[1] = distance;
    get_output_buffer()[2] = 0;
}
```

ultrasonic_module.h

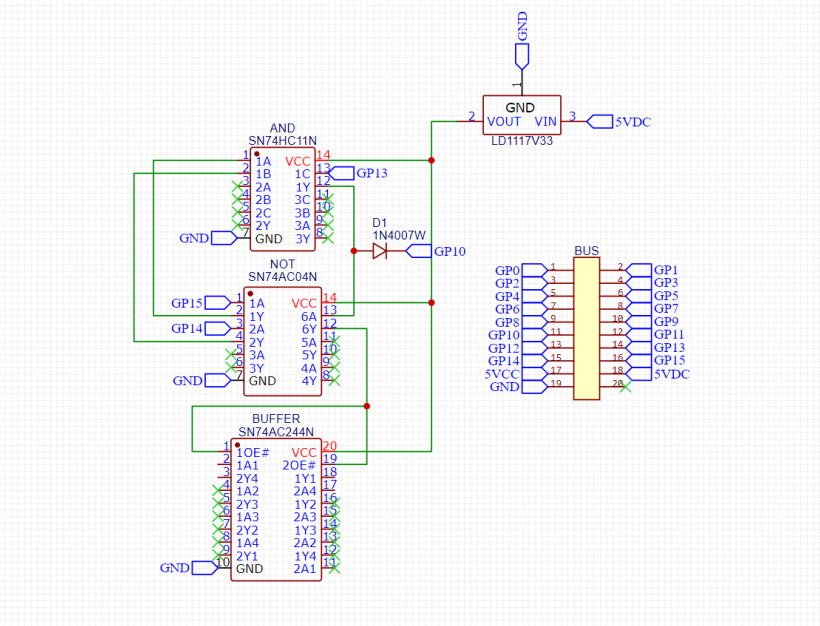
```
#ifndef ULTRASONIC_MODULE_H_
#define ULTRASONIC_MODULE_H_

#define TRIG 1
#define ECHO 0

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "pico/stdlib.h"
#include "hardware/gpio.h"
#include "hardware/timer.h"
#include "bluetooth.h"

void init_ultrasonic_module();
int get_distance();
void ultrasonic_module_reaction();

#endif // ULTRASONIC_MODULE_H_
```

bluetooth.c

```
void response() {
    gpio_put(25, uart_data_waiting);
    motor_driver(input_buffer[1]);
    switch (input_buffer[0]) {
    case 0:
        // neutral
        break;
    case 1:
        // ultrasonic_module
        ultrasonic_module_reaction();  // <- dodana funkcija za obradu podataka
        bluetooth_send();              // <- funkcija slanja podataka
        break;
    case 2:
        // matrix_module
        matrix_module_reaction();
        break;
    case 3:
        // ID nije korišten
        break;
    case 4:
        // ID nije korišten
        break;
    case 5:
```

```
        // ID nije korišten
        break;
    case 6:
        // ID nije korišten
        break;
    case 7:
        // demo_module
        demo_module_reaction();
        break;
    case 18:
        // rescan
        scan_for_modules();
        send_return_message();
        break;
    case 19:
        // Promjena modula
        set_module_id(input_buffer[2]);
        module_setup(input_buffer[2]);
        break;
    default:
        break;
    }
}
```

Stvaranje PCB-a



Ovako bi izgledao osnovni dio modula, Ovo je primjer za ID 1 (prouči NOT gate). Putem Buffera se kontroliraju komponente. 3 integrirana kruga kontroliraju protok podataka prema ostalim komponentama.

Marketing

Ovaj projekt nije stvoren s ciljem profitiranja, već s namjerom dijeljenja znanja i poticanja učenja. Za nas je ovo bilo iznimno važno iskustvo koje je omogućilo primjenu teorijskog znanja u praksi, ali isto tako želimo da bude korisno iskustvo i za druge.

Projekt će uvijek ostati besplatan i open-source, te se nadamo da će nastaviti rasti i razvijati se s vremenom. Nema potrebe za uvođenjem reklama jer naš fokus nije na stvaranju profita, a prisustvo reklama samo bi narušilo korisničko iskustvo.

Želimo da ovaj projekt ostane dostupan svima i nadamo se da će se širiti kroz uključivanje što većeg broja škola i pojedinaca zainteresiranih za stjecanje znanja.

Budućnost

U ovom projektu, stvarno je istina da je jedino ograničenje vaša mašta. Trudili smo se stvoriti sustav koji je jednostavno proširiv, otvarajući vrata beskonačnim mogućnostima.

Imamo nekoliko uzbudljivih ideja za budućnost projekta. Glavna je ideja stvoriti web stranicu koja bi funkcionirala kao platforma za razmjenu ideja i povezivanje ljudi zainteresiranih za razvoj modula. Na toj platformi, pojedinci mogu objaviti svoje ideje, a zainteresirani se mogu pridružiti timu kako bi ih zajedno razvili.

Osim toga, imamo i konkretnije ideje za razvoj novih modula, uključujući LCD modul, modul za praćenje kvalitete zraka, senzor praćenja svjetlosti te modul za robotsku ruku. Također razmišljamo o poboljšanju ultrazvučnog modula kako bi ROVER mogao autonomno navigirati i kretati se bez potrebe za stalnim nadzorom.

S ovim planovima, vjerujemo da će projekt i dalje rasti i razvijati se kako bi pružio još više mogućnosti za kreativnost i inovacije.

Zaključak

Ovaj projekt predstavljao je iznimno važno iskustvo za nas, pružajući nam priliku da naučimo nove stvari i unaprijedimo već postojeće vještine. Rad u timu pokazao se ključnim, jer smo stekli vrijedno iskustvo suradnje, što je bitna vještina za našu budućnost.

Sretni smo što smo imali priliku naučiti mnogo toga iz različitih područja industrije, što nam je omogućilo širu perspektivu i bolje razumijevanje kompleksnosti projekta. Nadamo se da će naš rad inspirirati i potaknuti druge da se uključe te doprinesu projektu svojim idejama i vještinama.

Na kraju, želimo se zahvaliti Strukovnoj Školi Vice Vlatkovića i svim njihovim djelatnicima na njihovoj podršci i poticaju koji su nam pružili tijekom ovog projekta.