

Avto na Daljinsko Upravo

Jonas Korene Novak, Matic Flegar
jonas.korene.novak@studnets.vegova.si
matic.flegar@students.vegova.si
Vegova Ljubljana
Razred R3D
20. maj 2026

Kazalo

1	Uvod	2
2	Predstavitev in Analiza Projekta	2
3	Težave in Rešitve	2
4	Končna Izvedba Projekta	3
4.1	Strojna Oprema	3
4.2	Programska Oprema - Definicije	3
4.2.1	Definirani Pini	3
4.2.2	Definirani Paketi	3
4.3	Programska Oprema	4
5	Zaključek	6

1 Uvod

Za svoj zaključni projekt pri predmetu IOT, v 3. letniku izobraževalnega programa *Tehnik Računalništva* na Vegovi, sva se odločila narediti enostavni **Avto na daljinsko upravo**.

To poročilo bo, v sledečih poglavjih predstavilo postopek izdelave, težave, rešitve, in morebitno kaj več.

2 Predstavitev in Analiza Projekta

Najin izbran projekt je bil avto na daljinsko, saj se nama je zdelo zanimivo, kako lahko narediva avto ki se dobro upravlja, z opremo ki jo imava na voljo.

Za svoj projekt sva uporabila **Raspberry Pi Pico W** kot svoj mikrokontroler, že-narejen PCB od **Freenove 4WD Car Kita** in **DC Motorje** za pogon.

Na začetku sva hotela spisati kodo za MCU (*Mikrokontroler - RPi Pico W*) v jeziku **C**, vendar sva zaradi hitre iteracije različic in male praktične razlike v hitrosti (saj hitrost procesiranja ni bila kritična) prepisala kodo v bolj berljivem **MicroPythonu**.

3 Težave in Rešitve

Med izvedbo projekta sva hitro naletela na eno težavo, in sicer, metodo upravljanja. Saj smo se to učili pri pouku, sva želela na začetku uporabiti **Web Interface** za upravljanje, serviran preko **HTTP-ja**, vendar je to razkrilo nekaj kritičnih težav:

1. HTTP temelji na **TCP-ju** (ang. *Transmission Control Protocol*). TCP je **povezaven protokol**, kar pomeni da morata klient (naš Web Interface) in strežnik (naš MCU) držati konstantno povezavo, kar pomeni za je čas za vzpostavljjanje povezave daljši.
2. (V kontekstu z točko 1) TCP je **počasen**. TCP ima glavo veliko **od 20 do 60 bajtov**, ki jih morata klient in strežnik dekodirati, preurediti v primeru zamešanega zaporedja (kar sicer ni teževa na močnem klientu, vendar naš strežnik - Pico W, ni ravno hiter), itd.
3. HTTP **ni narejen za take stvari**. HTTP je, kot ime pove, narejen za prenašanje **Hiperteksta** (ang. *Hypertext*) (torej spletnih strani), ne pa podatkov za kontrolo avta.

Zato sva se odločila **opustiti HTTP in TCP** in uporabiti **svoj binarni protokol**, ki temelji na **UDP** (ang. *User Datagram Protocol*). UDP ima sicer tudi svoje težave, vendar je za najino uporabo bolj primeren.

UDP in svoj protokol nam sicer **odstrani garancijo** TCP-ja da bodo podatki pravilno dostavljeni od A do B, vendar za našo uporabo, kjer pošiljamo 30 paketov na sekundo, je ena izguba vsakih par 100 paketov **zanemarljiva**. Dodatno nam uporaba svojega binarnega protokola tudi dopusti da imamo **popolno kontrolo** nad **vsebino paketa** in ne dobimo zraven še ogromno t.i. *overheada* od HTTP-ja.

Končno primerjavo lahko vidite tukaj:

Feature	TCP + HTTP	UDP + Bin. Protokol
Tip Povezave	Povezljiv	Nepovezljiv
Zanosljivost	Garantiran Prenos	<i>Best-Effort</i> Prenos
Hitrost	Počasen	Hiter
Preverjanje Napak	Da	Samo osnovni checksum
Vrstni Red Paketov	Garantiran	Ni Garantiran
Velikost Glave	20B do 60B	8B
Izkoriščenost Paketa	Mala	Popolna

4 Končna Izvedba Projekta

4.1 Strojna Oprema

- **MCU** - Raspberry Pi Pico W
- **PCB** - Freenove 4WD Car PCB
- **Pogon** - DC Motorji in Mecanum Kolesa
- **Napajanje** - 2x N18650CP Baterija

Komponente so povezane med sabo več ali manj, direktno.

4.2 Programska Oprema - Definicije

4.2.1 Definirani Pini

```

PIN_MOTOR_PWM_RIGHT3 = 9
PIN_MOTOR_PWM_RIGHT4 = 8
PIN_MOTOR_PWM_LEFT1  = 18
PIN_MOTOR_PWM_LEFT2  = 19
PIN_MOTOR_PWM_LEFT3  = 21
PIN_MOTOR_PWM_LEFT4  = 20

```

4.2.2 Definirani Paketi

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x00								NEUPORABLJENO																							
Left X																															
Left Y																															
Right X																															
Right Y																															

}

Kontrolni Paket

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x01								NEUPORABLJENO																							
IPv4 MCU-ja																															

}

Adv. Paket

4.3 Programska Oprema

Spisana v MicroPythonu, koda deluje v večih korakih.

1. Program se zažene preko funkcije `main()`.
2. Funkcija `main()` požene asinhroni strežnik z:

```
asyncio.run(run_server())
```
3. Funkcija `run_server()` najprej inicializira Wi-Fi povezavo:
 - (a) Aktivira WLAN v načinu odjemalca.
 - (b) Poskuša povezati napravo na podano SSID omrežje.
 - (c) Čaka največ 10 sekund na uspešno povezavo.
 - (d) Če povezava ne uspe, se program zaključi.
4. Po uspešni povezavi program inicializira motorje:
 - (a) Ustvari PWM izhode za vse štiri motorje.
 - (b) Vsak motor ima:
 - PWM za vožnjo naprej,
 - PWM za vožnjo nazaj.
5. Program nato ustvari UDP socket:
 - (a) Omogoči broadcast komunikacijo.
 - (b) Veže socket na naslov `0.0.0.0:8888`.
 - (c) Socket nastavi v neblokirni način.
6. Zažene se asinhrona naloga za periodično oglaševanje IP naslova:
 - (a) Vsako sekundo naprava pošlje broadcast UDP paket.
 - (b) Paket vsebuje:
 - tip paketa `0x01`,
 - lokalni IP naslov naprave.
7. Glavna zanka strežnika začne poslušati UDP pakete:
 - (a) Program neprestano kliče `recvfrom()`.
 - (b) Če ni podatkov:
 - program kratko počaka,
 - nato nadaljuje poslušanje.
8. Ko UDP paket prispe:
 - (a) Program prebere prvi bajt kot tip paketa.
 - (b) Če tip paketa ni podprt:

- izpiše opozorilo,
 - paket ignorira.
9. Če je paket tipa `INPUT_PACKET_TYPE`:
- (a) Program preveri velikost paketa.
 - (b) Paket mora vsebovati:
 - 1 bajt tipa,
 - 4 vrednosti `float32`.
10. Program dekodira vhodne osi joysticka:
- (a) Poskusi interpretacijo:
 - little-endian,
 - big-endian.
 - (b) Preveri, ali so vrednosti smiselne:
 - niso NaN,
 - niso neskončne,
 - niso izven dovoljenega območja.
11. Po uspešnem dekodiranju program pridobi:
- `left_x`,
 - `left_y`,
 - `right_x`,
 - `right_y`.
12. Program izvede “tank drive” logiko:
- (a) Leva palica določa hitrost leve strani vozila.
 - (b) Desna palica določa hitrost desne strani vozila.
 - (c) Y os se obrne.
 - (d) Na vhod se uporabi deadzone filter.
 - (e) Vrednosti se omejijo na območje od -1 do 1.
 - (f) Pretvorijo se v hitrost motorjev od -100 do 100.
13. Funkcija `set_motor_speed()`:
- (a) določi smer vrtenja,
 - (b) izračuna PWM intenziteto,
 - (c) aktivira forward ali reverse PWM.
14. Program neprestano nadaljuje sprejemanje novih UDP ukazov.
15. Če pride do prekinitve programa:
- (a) ustavi IP broadcast nalogo,

- (b) vse motorje nastavi na hitrost 0,
- (c) deaktivira PWM izhode,
- (d) zapre UDP socket,
- (e) program se zaključi.

5 Zaključek

Pri izdelavi projekta *Avto na Daljinsko Upravo* sva pridobila veliko praktičnega znanja iz področij programiranja mikrokontrolerjev, omrežne komunikacije in načrtovanja lastnih protokolov. Med delom sva se srečala z različnimi tehničnimi težavami, predvsem pri izbiri komunikacijskega protokola in optimizaciji odzivnosti upravljanja vozila.

S primerjavo različnih pristopov sva ugotovila, da je za sistem, kjer je pomembna hitrost in nizka zakasnitev, uporaba UDP protokola bistveno primernejša od HTTP komunikacije preko TCP-ja. Prav tako sva spoznala prednosti uporabe lastnega binarnega protokola, saj omogoča manjši *overhead*, hitrejšo obdelavo podatkov in večjo kontrolo nad vsebino paketov.

Projekt nama je omogočil tudi boljše razumevanje delovanja PWM signalov, asinhronnega programiranja v MicroPythonu ter komunikacije med klientom in strežnikom v realnem času. Poleg tehničnega znanja sva izboljšala tudi sposobnost odpravljanja napak, testiranja in sodelovanja pri razvoju večjega projekta.

Končni rezultat je funkcionalen daljinsko voden avtomobil, ki se odziva hitro in stabilno, hkrati pa predstavlja dobro osnovo za nadaljnje nadgradnje, kot so dodajanje senzorjev, kamere, avtonomne vožnje ali naprednejših načinov upravljanja.

S projektom sva uspešno dosegla zastavljene cilje ter pridobila veliko uporabnih izkušenj, ki nama bodo koristile pri nadaljnjem izobraževanju in prihodnjih projektih.