Integracija mikrokrmilnika ESP32 in LEGO mindstorms

Kevin Černeka
Mentor: Gregor Černe
Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana
kevin.cerneka@gmail.com, gregor.cerne@fe.uni-lj.si

Integration of microcontroller ESP32 and LEGO mindstorms

The article presents a proof of concept of the simple and time-effective mobile system development process, which lowers learning curve for new users in student competitions and reduced time used for communication and data analysis. Mobile system is constructed using Lego Mindstorms (motors, sensors, physical structure), which is controlled with microcontroller ESP32, programmed via Arduino enviroment. The data from experiments is logged and send via simple file transfer over wireless connection to the computer, where the data is used to create dynamical model and corresponding control laws using MATLAB. The development process was used in the development of mobile systems for driving in straight line.

Kratek pregled prispevka

Članek predstavlja koncept enostavnega in časovno nezamudnega procesa razvoja mobilnega sistema, ki zmanjšuje čas učenja za nove uporabnike v sklopu študentskih tekmovanj. Dodatno skrajša čas za vzpostavitev komunikacije in analize podatkov. Mobilni sistem je sestavljen iz LEGO Mindstorms komponent (motorji, senzorji, fizična struktura), katerega upravljamo s pomočjo mikrokrmilnika ESP32, programiranega v Arduino okolju. Podatki pridobljeni iz eksperimentov so zabeležni in z uporabo enostavnega prenosa datotek poslani preko brezžične povezave na računalnik, kjer se podatki uporabijo za izdelavo dinamičnega modela in izdelavo regulatorja procesa v programskem okolju MATLAB. Razvojni proces je bil uporabljen za izdelavo mobilnega sistema za vožnjo naravnost.

1. Uvod

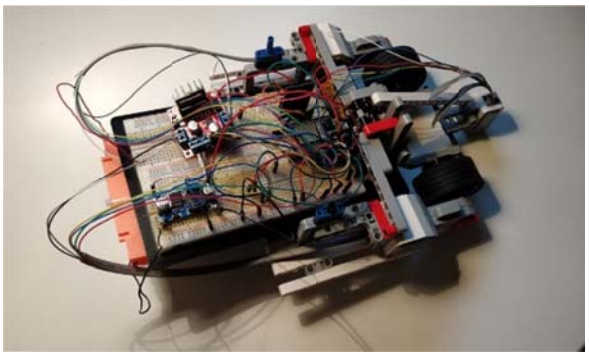
Pri procesu razvoja mobilnega sistema potrebujemo napraviti fizično konstrukcijo ter programsko kodo vodenja, hkrati pa želimo analizirati meritve z možnimi programskimi orodji. Paket LEGO Mindstorms EV3 omogoča hitro konstrukcijo robota saj ne potrebujemo vrtanja, žaganja, 3D modeliranja ipd.. Za programiranje manjših robotov je najbolj razširjena platforma Arduino, ki ima ogromno bazo uporabnikov zaradi enostavnega programiranja in uporabe. Namesto Arduino razvojne plošče se mnogokrat uporablja mikrokrmilnik ESP32. ESP32 je združljiv z razvojnim okoljem Arduino poleg tega ponuja tudi WiFi ter Bluetooth komunikacijo, dve jedri, strojno podprte SPI ter I2C povezave za le nekaj evrov. Za analizo podatkov pa se večinoma uporablja višje nivojska programska okolje kot so npr. MATLAB, Python, ipd. katera ponujajo veliko naprednih algoritmov za obdelovanje, prikaz in analizo podatkov.
Projekt služi kot dokaz koncepta procesa razvoja mobilnega sistema z združitvijo prednosti treh svetov: priročnost LEGO komponent, enostavnost programiranja v Arduino okolju zaradi zelo razširjene platforme ter ogromno knjižnic in nazadnje zmogljivega programskega okolja za analizo podatkov MATLAB. Cilj je konstrukcija robota kateri bo vozil v ravni črti.

2. Opis naprave

Mobilni sistem (naprej robot) je sestavljen iz mikokrmilnika ESP32, SD card modula, 5V regulatorja napetosti, 9V baterije, L298N mostiča za vodenje motorjev ter dveh LEGO motorjev EV3. Sestavljeni robot je prikazan na sliki 1.

2.1. Mikrokrmilnik ESP32

Mikrokrmilnik je najpomembnejši del saj predstavlja »možgane« našega robota. Mikrokrmilnik (prikazan na sliki 2) povezuje komponente in skrbi za delovanje robota (zajem in zapis podatkov, regulacija, …). Prednosti mikrokrmilnika ESP32 je več, omogoča WiFi in Bluetooth povezavo, 36 pin-ov, od tega 16 PWM kanalov ter možnost programiranja v Arduino okolju.



Slika 1: Robot.

Slika 2: Mikrokrmlink ESP32.

2.2. VS Code in PlatformIO

Za programiranje ESP32 je najbolj razširjena platforma Arudino, predvsem zaradi enostavnosti programiranja ter dostopa do veliko primerov programskih kod in že ustvarjenih knjižnic za LEGO komponentne. Za programiranje v Arduino okolju sem uporabljal program VS Code z dodatkom PlatformIO kateri podpira veliko število različnih mikrokrmilnikov in čipov. Poleg tega se enostavno dostopa do veliko knjižnic za različne platforme in namene.

2.3. LEGO EV3 motor

LEGO EV3 motor poganja enosmerna napetost. Motor krmilimo s pomočjo pulzno širinske modulacije (krajše PWM). Pulzno širinska modulacija je način krmiljenja motorjev pri katerem uporabljamo različno dolge signale vrednosti 1. Kadar je signal dolg celo periodo motor poganjamo z maksimalno močjo, v

nasprotnem primeru pa je moč motorja sorazmerna dolžini signala v eni periodi. Torej če je signal dolg polovico periode motor obratuje s polovico moči itn.. Poleg upravljanja s PWM-jem ima motor tudi optični senzor zasuka koles. ESP32 nam omogoča dvojno natančnost napram nativnemu enkoderju in sicer merimo zasuk s 0.5° natančnosti.

3. Opis sistema

Za lažje modeliranje je sistem predstavljen drugače. Namesto dveh vhodov v sistem kateri bi bile vrednosti dutycyclov (sorazmerna hitrost motorja) je samo en vhod označen s spremenljivko u_p. Delovna točka je že intergrirana znotraj sistema in sicer oba motorja imata za delovno točko nastavljeno vrednost dutycycla na 150. Z vhodom v sistem spreminjamo dutycycle desnega motorja (u_D) in levega motorja (u_L) kot je prikazuje enačba 1.

$$u_D = 150 + u_p$$
$$u_L = 150 - u_p \quad (1)$$

Izhod sistema ponazarja spremenljivka Δy. Vrednost spremenljivke Δy je razlika med vrednostmi zasuka levega (y_L) in desnega (y_D) motorja. Izračun vrednost Δy prikazuje enačba 2.
Takšen izhod je izbran za namen regulacije napake zasukov, če je napaka zasukov nič robot vozi naravnost.

$$\Delta y = y_L - y_D \quad (2)$$

3.1. Modeliranje

Za izvedbo identifikacije sistema je uporabljen stopničasti signal (z zelo barvo prikazan na sliki 3) različnih amplitud in dolžin, kateri služil kot vzbujanje sistema. Spremenljivka u_p zavzema vrednosti med +30 in -30. Glede na vhod spremenljivki u_D in u_L zavzemata vrednosti med 180 in 120. Želimo si veliko območje vzbujanja saj je tako razmerje signal šum manjše.

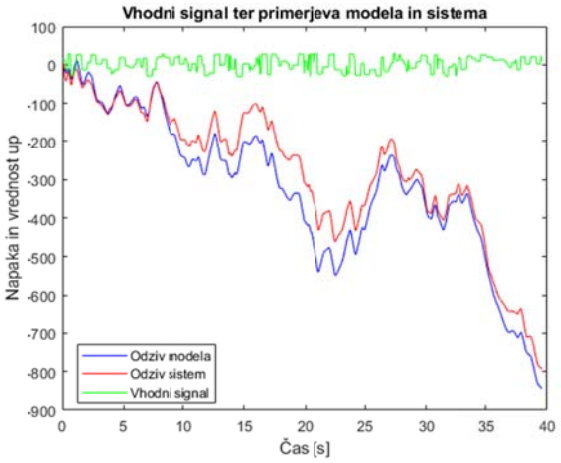
Med eksperimentom se podatki o vrednosti vhoda, zasukov in napake zapisujejo na SD kartico. Robot nato vzpostavi povezavo z WiFi omrežjem in ustvari strežnik preko katerega prenesemo datoteko s podatki na računalnik.

Na podlagi podatkov in metode najmanjših kvadratov (podrobneje opisana v [2]) dobimo diskretni model sistema (enačba 3).

$$G(z) = \frac{-0.1216z - 0.07364}{z^2 - 1.362z + 0.3616} \quad (3)$$

Diskretni sistem pretvorimo v zveznega s pomočjo zadrževalnika ničtega reda (angl. zero-order hold ali krajše ZOH). Zadrževalnik je podrobneje opisan [3]. Dobljeni zvezni model prikazuje enačba 4.

$$G(s) = \frac{-0.2456s - 124.4}{s^2 + 20.35s - 0.4174} \quad (4)$$



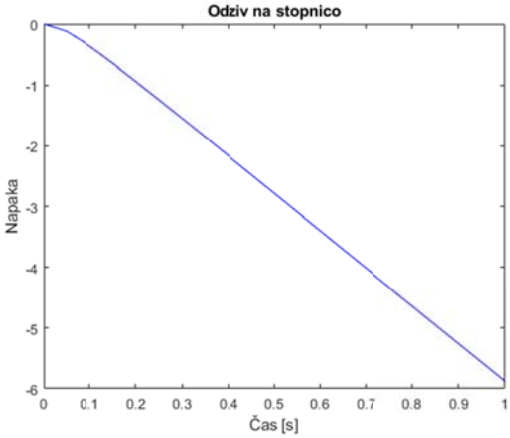
Slika 3: Vhodni signal ter odziv sistema in modela.

Slika 5 prikazuje tudi odziv modela in sistema na vhodni signal. Z rdečo barvo je prikazan odziv sistema na vhodni signal, z modro odziv zveznega modela na vhodni signal. Opazimo, da se model v določenih točkah zelo dobro ujema z sistemov, drugje nekoliko manj.

3.2. Regulator

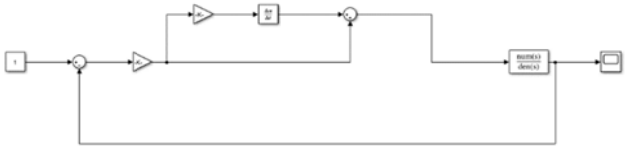
Regulator načrtujemo s simulacijo na pridobljenem modelu, saj tako zmanjšamo skupni čas načrtovanja. Simulacija poteka zelo hitro. Če simulacije ne bi uporabili bi to

pomenilo da vsakič na novo naložimo program na robota ga zaženemo in pridobimo podatke, nato popravimo parametre in ponavljamo postopek dokler ne pridobimo ustreznih parametrov regulatorja. To je seveda zelo zamudno. Za potrebe načrtovanja regulatorja je potrebno pridobiti odziv modela na enotino stopnico kateri je prikazan na sliki 4. Iz odziva je razvidno, da ima model negativno ojačanje in je intergrirnega značaja, saj izhod nenehno narašča. Če bi želeli, ima sistem pozitivno ojačanje bi morali pri izračunu Δy odšteti vrednost desnega zasuka od levega in ne vrednost levega od desnega zasuka.



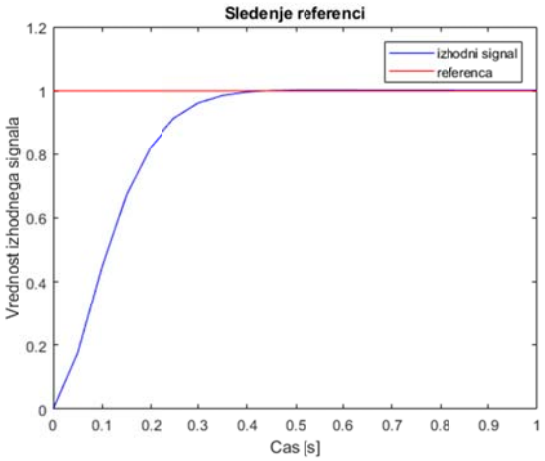
Slika 4: Odziv sistema na stopnico.

Na podlagi odziva na stopnico se s pomočjo Ziegler-Nicholsonove metode (podrobneje opisana v [1]) določi parametre regulatorja. Na podlagi regulatorja K_p in časovno konstanto T_D. Na podlagi Ziegler-Nicholsonove metode in preizkušanja različnih vrst regulatorjev se najbolje obnese regulator PD. Torej proporcionalni in diferencialni regulator. Slika 5 prikazuje realizacijo regulatorja v okolju SIMULINK.



Slika 5: PD regulator.

Sledenje regulatorja referenčni vrednosti 1 prikazuje slika 6. Iz odziva regulatorja je razvidno da je čas vzpona hiter, brez prenihaja ter z zelo majhnim pogreškom v ustaljenem stanju.



Slika 6: Sledenje referenci.

Zvezni regulator na robotu realiziramo s pomočjo enačbe 3, kjer u(k) predstavlja izhod regulatorja v trenutku k, u(k-1) izhod v trenutku k-1, ter e(k-i) za i zakasnjen pogrešek zasukov.

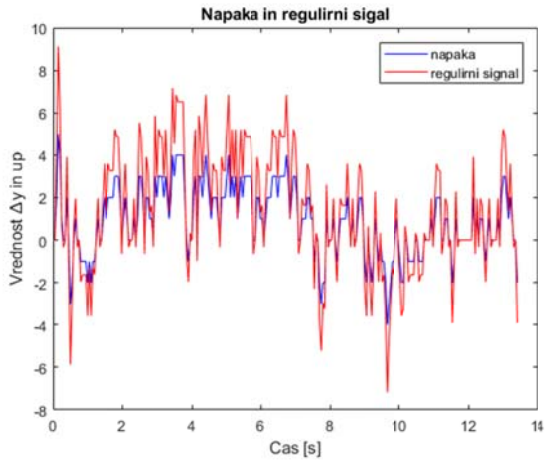
$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \quad (3)$$

Enačbe 4 pa prikazujejo pretvorbo parametrov zveznega regulatorja v koeficiente enačbe 3. Parameter T_0 predstavlja čas vzorčenja.

$$q_0 = K_p(1 + \frac{T_D}{T_0})$$
$$q_1 = -K_p(1 + 2\frac{T_D}{T_0}) \quad (4)$$
$$q_2 = K_p \frac{T_D}{T_0}$$

4. Rezultati

Končni rezultat je delujoči PD regulator za regulacijo napake zasukov kotov motorjev. Na sliki 7 je prikazan rezultat projekta. Z modro barvo je prikazan izhod sistema ob regulaciji napake, to je razlika med zasukoma levega in desnega motorja. Z rdečo je prikazan regulirni signal oz. izhod regulatorja.



Slika 7: Regulacija napake in regulirni signal.

Opazimo, da se napaka zasukov giblje med vrednostmi -4 in 4. Če želimo napako zasukov v ° moramo vrednost Δy prepoloviti, torej se napaka robota ob vožnji naravnost giblje med -2° in 2°. Pogrešek pri regulaciji pripisujem mehanski konstrukciji, saj kolesi nista idealno poravnani ter eno od koles ni najbolje pritrjeno zaradi obrabe motorja.

5. Zaključek

Rezultat projekta je dokaz koncepta, da lahko komponente LEGO Mindstorms učinkovito upravljamo s pomočjo mikrokrmilnika ESP32, zajemamo podatke jih s pomočjo brežžične povezave prenesemo na računalnik ter jih v programskem okolju MATLAB obdelamo, bodisi za namen modeliranja oz, evalvacije rezultatov.

Za pridobitev dobrega modela sistema je potrebno pravilno zastaviti sistem, izbrati bogate (frekvenčno, amplitudno) vhodne signali s katerimi pridobimo dinamiko sistema. Model dobro ponazarja dinamiko sistema, kar nam je omogočilo načrtati regulator, kateri dobro sledi referenci. MATALB nam je omogočil simulacijo modela s katerim smo skrajšali čas načrtovanja regulatorja.

6. Literatura

[1] *B. Zupančič, Avtomatsko vodenje sistemov*, Fakulteta za elektrotehniko v Ljubljani 2018.

[2] *S. Blažič, prosojnice pri predmetu IDENTIFIKACIJA,* Fakulteta za elektrotehniko v Ljubljani.

[3] *S. Blažič, Digitalno vodenje,* Fakulteta za elektrotehniko v Ljubljani 2013.

[4] *LEGO Group, "Developer Kits LEGO MINDSTORMS Education EV3",* Oktober 2018 [Elektronski] Dostopno: https://education.lego.com/enus/support/mindstorms-ev3/developer-kit.