»Mladi za napredek Maribora 2017«

34. srečanje

Krmiljenje avtomobilčka s pametnim telefonom

Raziskovalno področje: Elektrotehnika, elektronika

Raziskovalna naloga

Avtor: GAŠPER MAJAL
Mentor: MILAN IVIČ
Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA MARIBOR

Maribor, januar 2017

Vsebina

Kazalo slik

Kazalo tabel

1. POVZETEK

Raziskovalna naloga prikazuje krmiljenje avtomobilčka s pametnim telefonom Android. Avtomobilček poganjata dva enosmerna motorčka. S krmiljenjem teh dveh enosmernih motorčkov lahko avtomobilček vozi naprej, nazaj, lahko zavije v desno smer, v levo smer ali pa se ustavi. Motorčka krmili razvojna plošča Arduino Uno z Atmelovim mikrokontrolerjem ATmega 328P prek Bluetooth povezave s pametnim telefonom. Aplikacija za pametni telefon je izdelana v okolju MIT app inventorju.

Vmesnik med razvojno ploščo Arduino Uno in motorčkoma je integrirani H mostiček v gonilniku L293D. Z ustreznim krmiljenjem tega gonilnika lahko zagotovimo takšno delovanje obeh enosmernih motorčkov, da bo avtomobilček vozil po željah uporabnika.

Naloga prikazuje tudi delovanje in nastavitev časovnikov v mikrokontrolerju ATmega 328P. Oba enosmerna motorčka namreč krmilimo s pulzno širinsko moduliranim signalom, saj ne želimo da se motorčka vrtita prehitro oziroma prepočasi.

Izdelana je maketa avtomobilčka, ki prikazuje delovanje.

2. ZAHVALA

Zahvaljujem se vsem, ki so mi omogočili izdelavo te raziskovalne naloge, predvsem mentorju, brez katerega naloga nebi bila takšna, kot je. Mentorju se zahvaljujem za veliko novega znanja, ki mi ga je predal.

3. VSEBINSKI DEL

3.1 Uvod

Cilj raziskovalne naloge je izdelati avtomobilček, ki ga poganjata dva enosmerna motorčka. Krmiljenje avtomobilčka naj bo omogočeno s pametnim telefonom, za katerega bo izdelana aplikacija. Uporabljena bo Bluetooth povezava med avtomobilčkom in pametnim telefonom. Avtomobilček bo možno krmiliti na 5 različnih načinov:

- Vožnja avtomobilčka naprej.
- Vožnja avtomobilčka vzvratno.
- Zasuk avtomobilčka v levo.
- Zasuk avtomobilčka v desno.
- Zaustavitev avtomobilčka.

Avtomobilček se naj premika z ustrezno hitrostjo. V ta namen bo uporabljena pulzno širinska modulacija ustrezne frekvence in ustreznega duty cycla.

Cilj je spoznati delovanje časovnikov v mikrokontrolerju ATmega 328P, ki je srce razvojne plošče Arduino Uno. Najprej bojo predstavljeni glavni uporabljeni deli naloge, nato pa bo predstavljen potek raziskovanja, saj je bilo potrebno do končnega izdelka spoznati, raziskati in postopoma preizkusiti delovanje posameznih sklopov.

3.2 Enosmerni motorček

Enosmerni motorček v bistvu pretvarja električno energijo v mehansko. Ko na enosmerni motorček priključimo enosmerno napetost, se zaradi menjavanja smeri magnetnega polja vrti. Zgrajen je iz mirujočega, statorskega železnega jedra, na katerem se nahaja vzbujalno navitje za ustvarjanje magnetnega polja. Med magnetnimi poli statorja se nahaja rotor z navitjem, povezanim preko komutatorja in ščetk na zunanji vir enosmerne napetosti. Če menjamo polariteto priključene enosmerne napetosti, se motorček vrti v drugo smer.

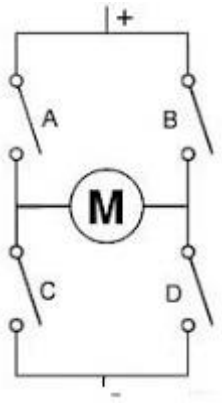
Slika 1: Enosmerni motorček (vir: Nanoelektronika).

Za spreminjanje polaritete priključene napetosti in s tem spreminjanje smeri vrtenja DC motorčka smo uporabili H-mostiček v integriranem vezju L293D.

3.3 Gonilnik L293D

H-mostiček:

Princip delovanja vezja, po katerem ima H-mostiček ime vidimo na sliki 2.



Slika 2: Prikaz vezja za spreminjanje polaritete napetosti na priključkih DC motorčka, H-mostiček (vir: avtor naloge).

Vezje H-mostička vsebuje štiri stikala: A, B, C in D. S preklapljanjem teh stikal enosmerni motorček krmilimo na različne načine:

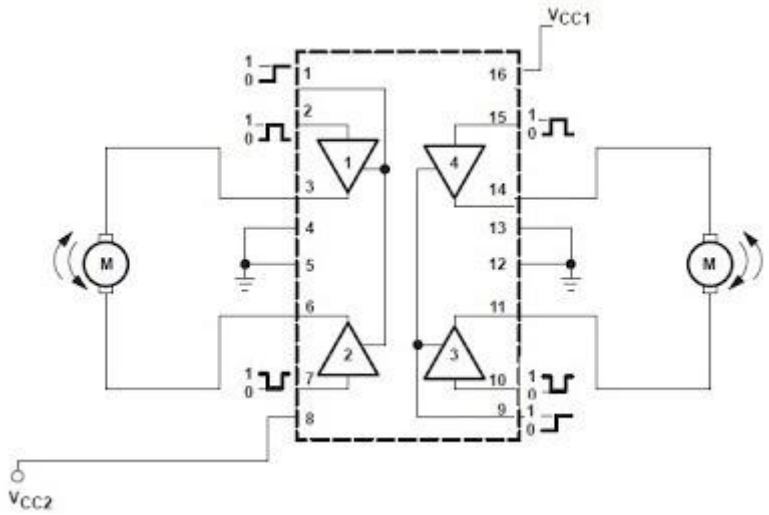
- Če vklopimo stikalo A in stikalo D, se bo motorček vrtel v smeri urinega kazalca,
- Če vklopimo stikalo B in stikalo C, se bo motorček vrtel v nasprotni smeri urinega kazalca,
- Če vklopimo stikalo A in stikalo B, se bo vrtenje motorčka ustavilo,
- Če vsa stikala izklopimo, bo motorček brez napajalne napetosti,
- Če istočasno vklopimo stikalo A in stikalo C ali stikalo B in stikalo D, je vezje v kratkem stiku, zato tega ne počnemo.

H-mostiček lahko izdelamo z različnimi elementi kot so releji, MOSFET, FET ali bipolarnimi BJT tranzistorji. Če pa naše zahteve po tokovni porabi niso prevelike, če želimo krmiliti majhne DC[1] motorčke, ki ne potrebujejo večjih tokov, lahko enostavno uporabimo integrirano vezje L293D.

Integrirano vezje L293D:

Integrirano vezje L293D je namenjeno za krmiljenje dveh enosmernih motorčkov, da se lahko vrtita v obe smeri. Mi bomo krmilili dva enosmerna motorčka. Nanj lahko priključimo motorček, ki za svoje delovanje ne potrebuje toka, večjega od 1 A (L293D 600 mA) in za napetosti od 4,5 V do 36 V. Vsi vhodi so združljivi s TTL[2] signali. Za zaščito ima L293D vgrajene hitre diode, ki ščitijo integrirano vezje pred napetostnimi konicami, ki nastanejo pri vklopu in izklopu motorčka (predvsem pri izklopu). Če se L293D segreje čez mejo 70 °C, vgrajeni senzorji ustavijo delovanje motorčka.

[1] DC (angl. Direct current – enosmerni tok).
[2] TTL (angl. Transistor transistor logic – družina digitalnih integriranih vezij).



Slika 3: Priključitev DC motorčkov na L293 (vir: Texas Instruments).

Levi motorček bomo priključili na priključka 3 in 6, krmilili pa ga bomo s priključki 1, 2 in 7. Desni motorček bomo priključili na priključka 11 in 14, krmilili pa ga bomo s priključki 9, 10 in 15.

Na priključek 8 (V_{CC2}) priključimo napetost, ki jo potrebuje enosmerni motorček za delovanje, na priključek 16 (V_{CC1}) pa napetost za gonilnik L293D, ki znaša 5 V.

Poglejmo delovanje posameznega DC motorčka v odvisnosti od logičnih nivojev na krmilnih priključkih gonilnika L293D.

Tabela 1: Delovanje DC motorčka, priključenega na levo stran gonilnika.

Priključek 1 (EN)	Priključek 2	Priključek 7	Funkcija
1	0	1	Vrtenje motorčka v smeri urinega kazalca.
1	1	0	Vrtenje motorčka proti smeri urinega kazalca.
1	0	0	Motorček se ne vrti.
1	1	1	Motorček se ne vrti.
0	neuporabno	neuporabno	Motorček se ne vrti.

Tabela 2: Delovanje DC motorčka, priključenega na desno stran gonilnika.

Priključek 9 (EN)	Priključek 10	Priključek 15	Funkcija
1	0	1	Vrtenje motorčka v smeri urinega kazalca.
1	1	0	Vrtenje motorčka proti smeri urinega kazalca.
1	0	0	Motorček se ne vrti.
1	1	1	Motorček se ne vrti.
0	neuporabno	neuporabno	Motorček se ne vrti.

Leva stran gonilnika L293:

- Priključek 1 je enable priključek. Povezan morata biti na +5 V (logična 1), če želimo da se bo motorček vrtel. Če želimo spreminjati hitrost vrtenja motorčka, na ta priključek pripeljemo PWM[3] signal.
- Priključka 4 in 5 morata biti povezana na skupno maso (GND[4]).
- Priključka 2 in 7 sta vhodna priključka gonilnika L293D in sta kontrolna priključka. Nadzorujeta delovanje motorčka.
- Priključka 3 in 6 sta izhodna priključka gonilnika L293D, kamor priključimo DC motorček.

Desna stran gonilnika L293:

- Priključek 9 je enable priključek. Povezan morata biti na +5 V (logična 1), če želimo da se bo motorček vrtel. Če želimo spreminjati hitrost vrtenja motorčka, na ta priključek pripeljemo PWM signal.
- Priključka 12 in 13 morata biti povezana na skupno maso (GND).
- Priključka 10 in 15 sta vhodna priključka gonilnika L293D in sta kontrolna priključka. Nadzorujeta delovanje motorčka.
- Priključka 11 in 14 sta izhodna priključka gonilnika L293D, kamor priključimo DC motorček.

Priključek 8 je priključek, kamor priključimo pozitivno napetost iz drugega napajalnika oziroma baterije za napetost DC motorčka (od 4,5 V do 36 V, odvisno od uporabljenega motorčka). Za naša motorčka smo uporabili baterijo 9 V.

Priključek 16 je namenjen za napetostno napajanje gonilnika L293D. Priključen mora biti na napetost +5 V.

3.4 Modul bluetooth HC-06

Bluetooth je brezžična tehnologija, namenjena predvsem mobilnim napravam, ki imajo baterijsko napajanje. Omogoča brezžično povezovanje med katerimikoli električnimi napravami, ki podpirajo ta standard. Bluetooth komunicira z drugimi napravami na frekvenčnem pasu okoli 2,45 gHz. Deluje na kratke razdalje do 10 metrov. Ker oddaja zelo majhno moč, do bistvenega segrevanja človekovega tkiva ne more priti.

Če želimo vzpostaviti povezavo z napravo Bluetooth, moramo vklopiti radijski vmesnik za Bluetooth v pametnem telefonu (Android). Ko prvič uporabimo novo napravo Bluetooth, jo moramo "združiti" z napravo, s katero jo želimo uporabljati, tako da obe napravi vesta, kako se varno povezati druga z drugo. Ko to naredimo, se samodejno povežeta.

[3] PWM (angl. Pulse width modulation – pulzno širinska modulacija).
[4] GND (angl. Ground – zemlja, masa).

Slika 4: Modul bluetooth (vir:dx.com)

Modul bluetooth HC-06 vsebuje štiri priključke, uporabili smo tri:

- Vcc, napetostno napajanje 3,3 V.
- GND, masa.
- Priključek TX. Priključek je namenjen sprejemanju podatkov. Ta priključek smo povezali s priključkom RX na razvojni plošči Arduino Uno. Preden zapisujemo programsko kodo na Arduino ploščo, prekinemo to povezavo, v nasprotnem primeru program ne bo uspešno zapisan.

3.5 Arduino Uno

Arduino Uno je razvojna plošča z mikrokontrolerjem ATmega 328P. Razvojna plošča vsebuje 14 digitalnih vhodno/izhodnih priključkov, med katerimi je 6 takšnih, ki jih lahko uporabljamo za PWM izhode. Vsebuje še 6 analognih vhodnih priključkov. Hitrost delovanja narekuje 16 MHz oscilator.

Programiramo ga s prosto dostopnim razvojnim okoljem Arduino IDE. Z računalnikom komunicira prek USB priključka.

Glavne značilnosti:

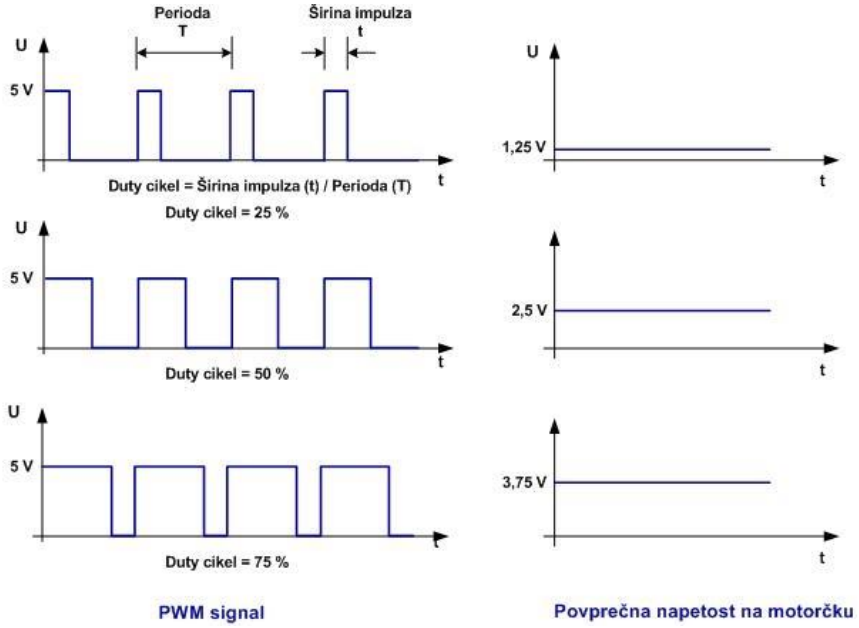
- Mikrokontroler ATmega 328P.
- Napajalna napetost 7 V - 12 V.
- Posamezni digitalni izhod lahko obremenimo s tokom do 20 mA.
- Priključek s 3,3 V izhodom lahko obremenimo s tokom do 50 mA.
- Pomnilnik 32 KB.
- Takt procesorja 16 MHz.

3.6 Preizkusi krmiljenja DC motorčka s PWM signalom

3.6.1 PWM signal

Najprej bomo en DC motorček krmilili z ustreznimi impulzi določene frekvence oziroma s pulzno širinsko moduliranim signalom. Poglejmo kaj je pulzno širinsko modulirani signal.

PWM signal je v bistvu sestavljen iz pravokotnih impulzov konstantne frekvence. Razmerje širine impulza proti periodi imenujemo duty cikel in je izraženo v odstotkih. PWM duty cikel, izražen v odstotkih, ustreza povprečni moči tega pulznega signala. Če PWM signal priključimo na enosmerni motorček, lahko reguliramo njegovo hitrost vrtenja. Pri različnem duty ciklu bo tudi hitrost vrtenja enosmernega motorčka različna.



Slika 5: PWM signal in povprečna vrednost napetosti na motorčku (vir: avtor naloge).

3.6.2 Hitrost vrtenja DC motorčka pri različnih vrednostih duty cycle PWM signala

Spreminjanje hitrosti vrtenja DC motorčka smo preverjali z različnima duty cycle PWM signala. Najprej smo morali ugotoviti, kako pridemo do PWM signala na razvojni plošči Arduino Uno. Zato smo morali spoznati vlogo in delovanje časovnikov (timerjev) v mikrokontrolerju.

Časovniki so srce avtomatizacije. Omogočajo nam nadzor ne le nad tem kaj se zgodi temveč kaj se dogaja. Omogočajo nam nadzor nad časom.

Časovniki ali bolj natančno časovniki / števci so del strojne opreme, vgrajene v Arduino Uno. Z njimi lahko merimo časovne dogodke, lahko pa z njimi štejemo impulze.

Arduino Uno vsebuje mikrokontroler ATmega 328P. Vsebuje tri časovnike, Timer0, Timer1 in Timer2. Timer0 in Timer2 sta 8-bitna, Timer1 pa je 16-bitni. Osnovna razlika med 8-bitnim in 16-bitnim je ločljivost. 8-bitni ima ločljivost 255, 16-bitni pa 65535. Timer0 je privzeto namenjen zakasnitvam (delay), timer1 krmiljenju servo motorčkov, timer2 pa generiranju tonov. Za generiranje PWM signala bomo uporabili timer2.

Vsi trije timerji so odvisni strojne ure. Arduino Uno poganja strojna ura (oscilator) frekvence 16 MHz. Privzeto so v Arduino Uno nastavljeni (konfigurirani) vsi timerji na frekvenco 1 KHz, privzeto pa so omogočene tudi prekinitve (interrupts).

3.6.2.1 Kako deluje časovnik (timer2):

Timer2 s svojim registrom TCNT2 (8-bitni, maksimalna vrednost 255) neprestano povečuje svojo vrednost za 1. Ko doseže maksimalno vrednost se postavi na 0 in začne od začetka, od 0. Takrat pravimo, da je prišlo do prekoračitve njegove vrednosti (overflow). Ob tem vklopi zastavico da vemo, da je prišlo do njegove prekoračitve. Vrednost zastavice lahko kontroliramo, lahko pa pri tem timer2 sproži prekinitveno rutino ISR (Interrupt Service Routine), če smo jo omogočili. V prekinitveni rutini se bo ponastavila zastavica, ki se je vklopila ob prekoračitvi vrednosti timerja2.

Da se lahko timerjeva vrednost povečuje, mora imeti dostop do ure (oscilator), ki konstantno generira signale. Vsakokrat ko timer2 zazna signal ure, se mu vrednost poveča za 1.

Najmanjša izmerljiva enota časa tega signala je ena perioda te ure. Če na primer uporabimo uro s frekvenco 1 MHz, lahko izračunamo čas trajanje ene periode:

$$T = \frac{1}{f} = \frac{1}{1\ MHz} = 1\ \mu s$$

Arduino Uno vsebuje kristalni oscilator vrednosti 16 MHz.

$$T = \frac{1}{f} = \frac{1}{16\ MHz} = 0{,}0625\ \mu s$$

Timer2 se torej poveča za 1 vsakih 0,0625 µs.

Timer2 ima svoje izhodne PWM priključke na priključku 11 (OC2A) in na priključku 3 (OC2B) razvojne plošče Arduino Uno. Njegova vrednost se, kot smo ugotovili, poveča za 1 vsakih 0,0625 µs. Maksimalno vrednost doseže v 16 µs (0,0625 x 256). Povečevanje vrednosti timerja2 lahko upočasnimo s preddelitvijo. Če hočemo timer2 uporabiti, ga moramo pravilno nastaviti. Timer2 vsebuje številne registre s katerimi nastavimo (določimo) njegovo delovanje. Dva od teh registrov, s katerima določimo njegovo delovanje sta TCCR2A in TCCR2B.

Tabela 3: Register TCCR2A (timer/counter control register).

COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20

Tabela 4: Register TCCR2B (timer/counter control register).

FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20

Tabela 5: Register TIFR (timer/counter interrupt register).

OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

Tabela 6: Načini delovanja (mode) timerja2.

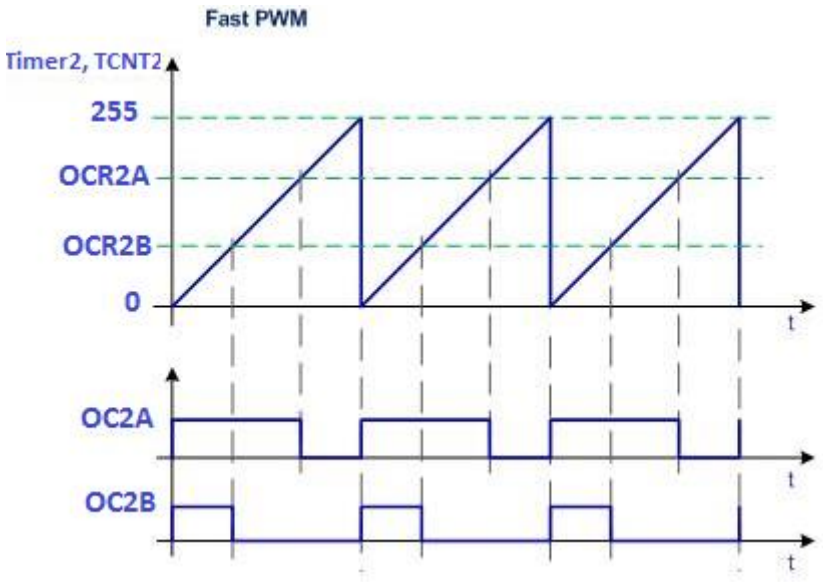
Mode	WGM22	WGM21	WGM20	Top	Opis:
0	0	0	0	0xFF	Normal
1	0	0	1	0xFF	PWM Phase Correct
2	0	1	0	OCRA	CTC
3	0	1	1	0xFF	Fast PWM
4	1	0	0	-	Rezervirano
5	1	0	1	OCR0A	PWM Phase Correct
6	1	1	0	-	Rezervirano
7	1	1	1	OCR0A	Fast PWM

Tabela 7: Vloga CS (clock select) bitov, preddelitev.

CS22	**CS21**	**CS20**	**Preddelitev:**
0	0	0	Timer2 stop
0	0	1	Preddelitev 1:1
0	1	0	Preddelitev 1:8
0	1	1	Preddelitev 1:32
1	0	0	Preddelitev 1:64
1	0	1	Preddelitev 1:128
1	1	0	Preddelitev 1:256
1	1	1	Preddelitev 1:1024

Mokrokontroler ATmega 328P vsebuje tudi preddelilnik, ki deli prožilne impulze namenjene časovniku Timer2. S preddelilnikom dosežemo, da se Timer2 povečuje po osmih, dvaintridesetih, ... ali po vsakih 1024-tih strojnih ciklih. Preddelilnik je torej koristen, kadar želimo s časovnikom Timer2 meriti daljše časovne intervale. Preddelilnik nastavljamo s prvimi tremi biti (CS20, CS21 in CS22) v registru TCCR2B.

Izmed načinov delovanja timerja2 bomo izbrali mode 3, **Fast PWM** način. Fast PWM imenujemo, ker lahko generiramo pulzno širinske signale visokih frekvenc. Timer2 povečuje svojo vrednost od 0 do 255. V registra OCR2A in OCR2B lahko zapišemo vrednosti od 0 do 255. Ko timer2 doseže vrednost, zapisano v OCR2A, na priključku 11 (OC2A) preklopi logično stanje iz 1 na 0. Ko doseže vrednost, zapisano v OCR2B, na priključku 3 (OC2B) preklopi logično stanje iz 1 na 0. Kadar pa pride na timerju2 do prekoračitve svoje vrednosti (ovwrflow), na obeh priključkih preklopi logično stanje iz 0 na 1.



Slika 6: Prikaz delovanja časovnika Timer2 v načinu Fast PWM, mode 3 (vir: avtor naloge).

Na priključkih 11 in 3 razvojne plošče Arduino Uno dobimo PWM signal, katerega frekvenca je odvisna od časa, v katerem se timer2 poveča od 0 do 255, duty cycle pa je odvisen od vrednosti registrov OCR2A in OCR2B. Upoštevati moramo, da se pri prekoračitvi vrednosti časovnika timer2 porabi dodaten strojni cikel. En strojni cikel moramo prišteti tudi pri izračunu duty cycle, ko preklopi logično stanje na OC2A oziroma OC2B.

Primer:

Izberimo preddelitev 1:1024. Pri 16 MHz oscilatorju je frekvenca PWM signala:

$$f_{PWM} = \frac{f_{osc}}{256 \cdot 1024} = 61\ Hz$$

V register OCR2A zapišimo vrednost 160. Izračunajmo duty cycle:

$$duty\ cycle = \frac{160 + 1}{256} \cdot 100\ \% = 63\ \%$$

V register OCR2B zapišimo vrednost 30. Izračunajmo duty cycle:

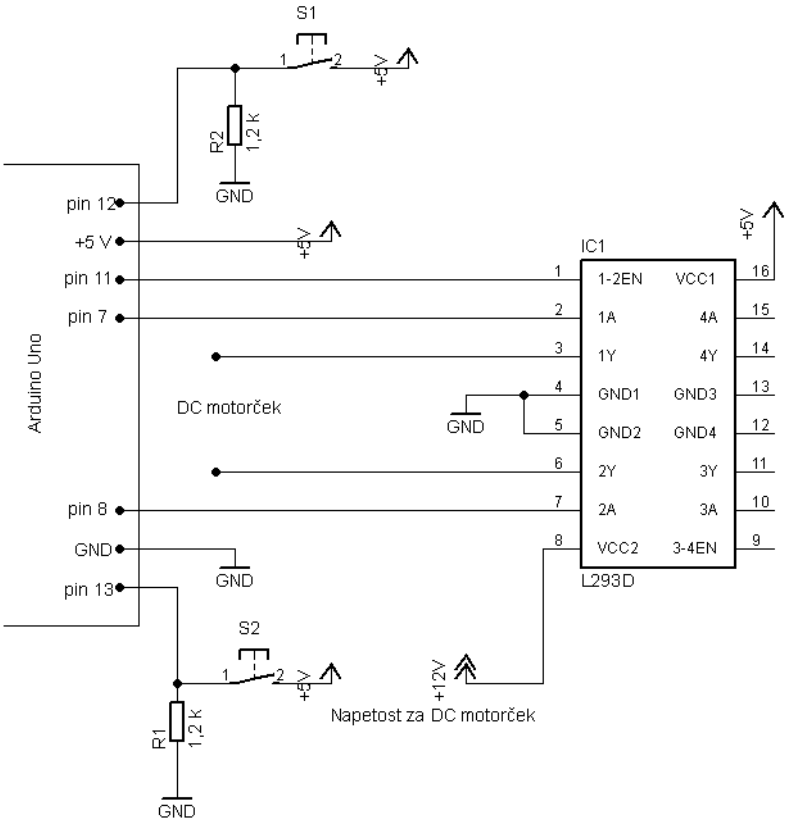
$$duty\ cycle = \frac{30 + 1}{256} \cdot 100\ \% = 12\ \%$$

Duty cycle PWM signala torej lahko spreminjamo s spreminjanjem vrednosti OCR2A in OCR2B. Vrednost, ki smo jo zapisali v register OCR2A bo vplivala na obliko signala na priključku 11, vrednost,

ki smo jo zapisali v register OCR2B pa bo vplivala na obliko signala na priključku 3 razvojne plošče Arduino Uno.

3.6.2.2 Praktični preizkus krmiljenja DC motorčka, primer 1:

Krmilili smo en DC motorček, priključen na levo stran gonilnika L293D. Na Arduino Uno (slika 7) smo priključili dve tipki. Če sklenemo prvo tipko (S1), se motorček vrti hitreje (OCR2A = 160), če sklenemo drugo tipko (S2), se motorček vrti počasneje (OCR2B = 30). Če ni nobena od tipk sklenjena ali pa sta sklenjeni obe tipki, se motorček ne vrti.



Slika 7: Vezje za preizkus delovanja, primer 1 (vir: avtor naloge).

Programska koda:

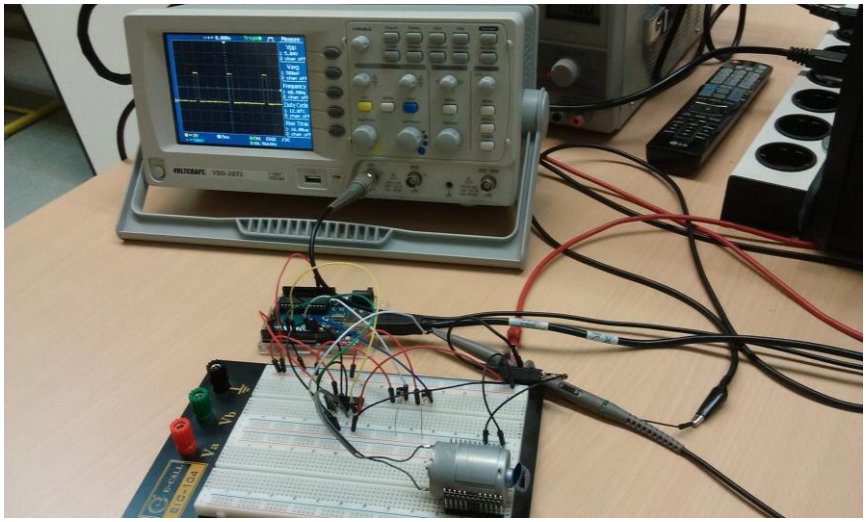
```
/*
 * Preizkus spreminjanja hitrosti vrtenja DC motorčka z različnima duty cycle PWM signala.
 * Primer => en DC motorček krmilimo z dvema tipkama. Če je sklenjena prva tipka, se motor
 * vrti hitreje (duty cycle večji), če je sklenjena druga tipka, se motor vrti počasneje (duty
 * cycle manjši). Frekvenca je konstantna 61 Hz.
 * Timer2 => f=61 Hz
 * mode 3 => Fast PWM
 * preddelitev 1:1024
 * Priključka: OCR2A => pin 11
 *             OCR2B => pin 3
 */
```

```
 //Pini za gonilnik L293D:
 #define M_desniA 7    //Signal za desni motorček, priključen na pin 2 na L293.
 #define M_desniB 8    //Signal za desni motorček, priključen na pin 7 na L293.
 #define M_desniE 3    //Enable pin za desni motorček, priključen na pin 1 na L293, PWM.

 int Tipka = 12;
 int Tipka1 = 13;
 int Polozaj_tipke = 0;
 int Polozaj_tipke1 = 0;

//Spremenljivki za PWM krmiljenje hitrosti vrenja DC motorčka (od 0 do 255):
int Enable1;
int Enable2;

void setup()
{
 //Določitev vhodov in izhodov:
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(Tipka, INPUT);
 pinMode(Tipka1, INPUT);

 Serial.begin(9600);

 //Sprememba frekvence PWM signala na Timer2 (mode 3 => FAST PWM, f=61 Hz):
 TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
 TCCR2B = _BV(CS20) | _BV(CS21) | _BV(CS22);                    //Preddelitev 1:1024
}

void loop()
{
 Polozaj_tipke = digitalRead(Tipka);
 Polozaj_tipke1 = digitalRead(Tipka1);

 if((Polozaj_tipke == HIGH) && (Polozaj_tipke1 == LOW))
 {
  OCR2A = 160;
  OCR2B = 30;

  Enable1 = OCR2A;
  Enable2 = OCR2B;

  analogWrite(M_desniE, Enable1);
  digitalWrite(M_desniA, HIGH);          //Vrtenje naprej, duty cycle 63 %.
  digitalWrite(M_desniB, LOW);

  Serial.print("Enable1: ");
  Serial.print(Enable1);
  Serial.print("  Enable2: ");
  Serial.println(Enable2);
```

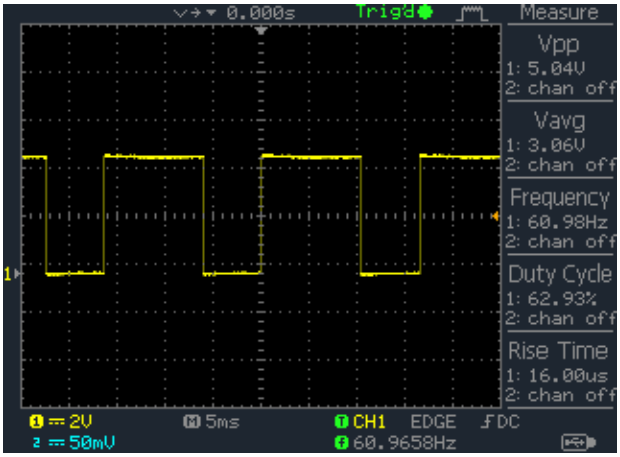
```
  }

  else
  {
   if((Polozaj_tipke == LOW) && (Polozaj_tipke1 == HIGH))
   {
    OCR2A = 30;
    OCR2B = 160;

    Enable1 = OCR2A;
    Enable2 = OCR2B;

    analogWrite(M_desniE, Enable1);
    digitalWrite(M_desniA, HIGH);              //Vrtenje naprej, duty cycle 12 %.
    digitalWrite(M_desniB, LOW);

    Serial.print("Enable1: ");
    Serial.print(Enable1);
    Serial.print("  Enable2: ");
    Serial.println(Enable2);
   }
   else
   {
    digitalWrite(M_desniA, LOW);               //Motor stop.
    digitalWrite(M_desniB, LOW);

    Serial.println("Motor stop");
   }
  }
  delay(10);
}
```

Glavne značilnosti programske kode so opisane v komentarjih. Ker vrednosti registrov OCR2A in OCR2B ne moremo direktno vpisati na ustrezni izhodni priključek, smo deklarirali spremenljivki z imenom Enable1 in Enable2. Z if stavkoma preverjamo stanje tipk. Izvede se blok ukazov tistega if stavka, katerega pogoj je resničen. Zanka void loop(), ki se neprestano ponavlja, ob vsaki ponovitvi preverja stanje tipk.

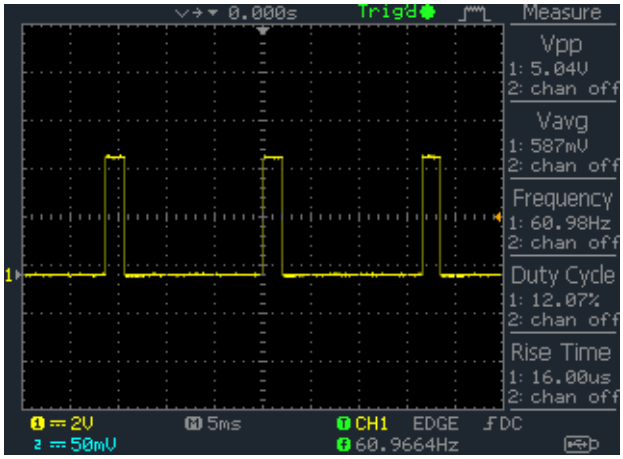
DC motorček smo priključili po načrtu iz slike 7 in preizkusili delovanje. Na priključek enable gonilnika L293D (priključek 1), povezanega s priključkom 11 Arduino Uno smo priključili digitalni osciloskop in preverili, ali se naši izračuni ujemajo z dejanskim stanjem. V obeh primerih so se izračuni ujemali, kar prikazujeta tudi oscilograma.



Slika 8: Preizkus delovanja z enim DC motorčkom in meritve (vir: avtor naloge).



Slika 9: PWM signal na enable pinu gonilnika L293D, sklenjena prva tipka, S1 (vir: avtor naloge).



Slika 10: PWM signal na enable pinu gonilnika L293D, sklenjena druga tipka, S2 (vir: avtor naloge).

Frekvenco PWM signala smo z ustrezno nastavitvijo bitov CS0, CS1 in CS2 v registru TCCR2B znižali na vrednost 61 Hz, saj se DC motorček pri nižjih obratih (manjši duty cycle) boljše obnaša pri frekvenci 61 Hz kot pri višjih frekvencah. V programu smo vse tri bite postavili na 1 in s tem dosegli preddelitev 1:1024 (timer2 poveča vrednost za 1 po 1024-tih strojnih ciklih).

3.6.2.3 Praktični preizkus krmiljenja DC motorčka, primer 2:

Krmilili smo dva DC motorčka, priključena na gonilnik L293D. Delovanje smo preizkušali s pomočjo dveh tipk (slika 11). Zaradi preglednosti povezav je razvojna plošča Arduino Uno razdeljena na dva dela.

Če sklenemo prvo tipko (S1), se desni motorček vrti hitreje (OCR2A = 200), levi motorček pa počasneje (OCR2B = 40). Duty cycle za krmiljenje desnega motorčka znaša 78 %, za krmiljenje levega pa 16 %.

Če sklenemo drugo tipko (S2), desni motorček vrti počasneje (OCR2A = 40), levi motorček pa hitreje (OCR2B = 200). Duty cycle za krmiljenje desnega motorčka znaša 16 %, za krmiljenje levega pa 78 %.

Če ni nobena od tipk sklenjena ali pa sta sklenjeni obe tipki, se motorček ne vrti.

Frekvenco PWM signala smo pustili nespremenjeno 61 Hz. Vrednosti OCR2A in OCR2B smo spremenili, da lahko z digitalnim osciloskopom preverimo, ali izračunani duty cycle ustrezajo dejanskim.

V register OCR2A zapišimo vrednost 200. Izračunajmo duty cycle:

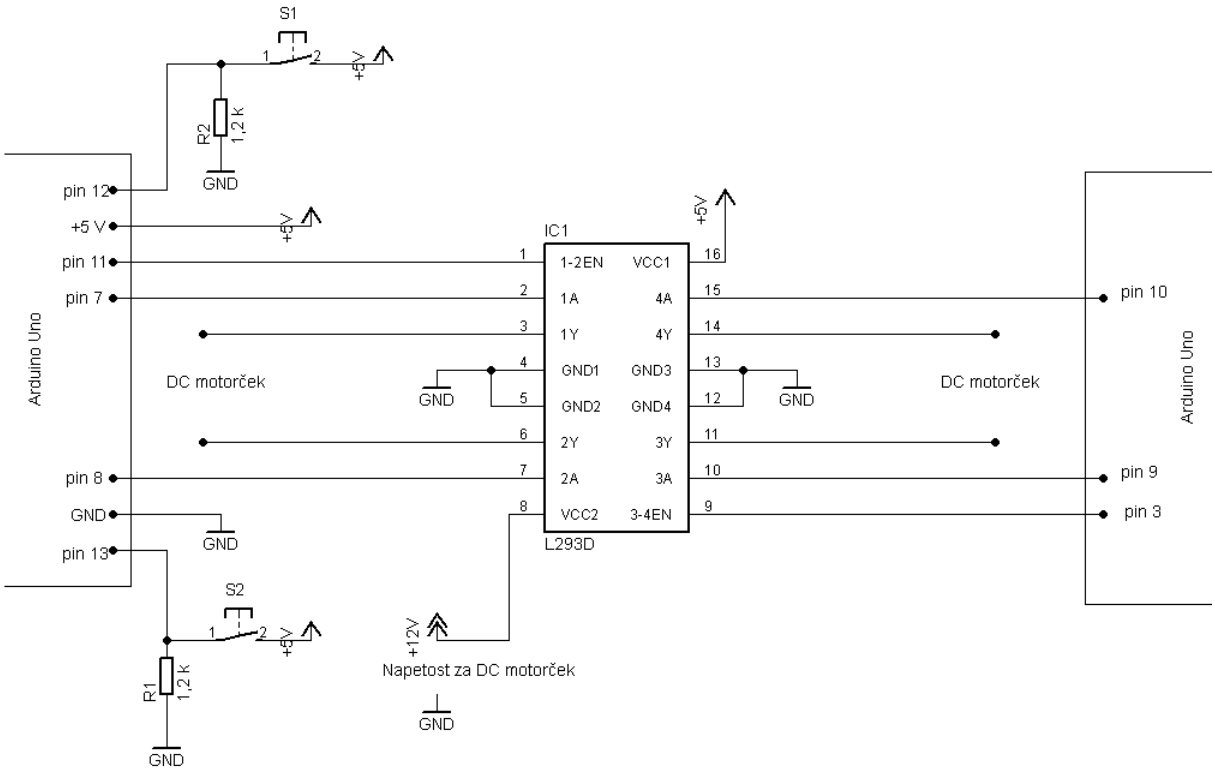
$$duty\ cycle_1 = \frac{200 + 1}{256} \cdot 100\ \% = 78{,}5\ \%$$

V register OCR2B zapišimo vrednost 40. Izračunajmo duty cycle:

$$duty\ cycle_2 = \frac{40 + 1}{256} \cdot 100\ \% = 16\ \%$$

V programsko kodo smo vključili serijsko komunikacijo (9600 bit/s) z namenom, da lahko med delovanjem na serijskem monitorju spremljamo vrednosti OCR2A in OCR2B. Ti dve vrednosti se namreč zapisujeta na enable priključka (1 in 9) gonilnika L293D.

Podatki se na serijski monitor izpisujejo v odvisnosti od tega, katera tipka je sklenjena. Če ni sklenjena nobena ali pa ste sklenjeni obe tipki, se DC motorčka ustavita, kar se tudi izpisuje na serijskem monitorju. Primere izpisa na serijskem monitorju prikazuje slika 12.



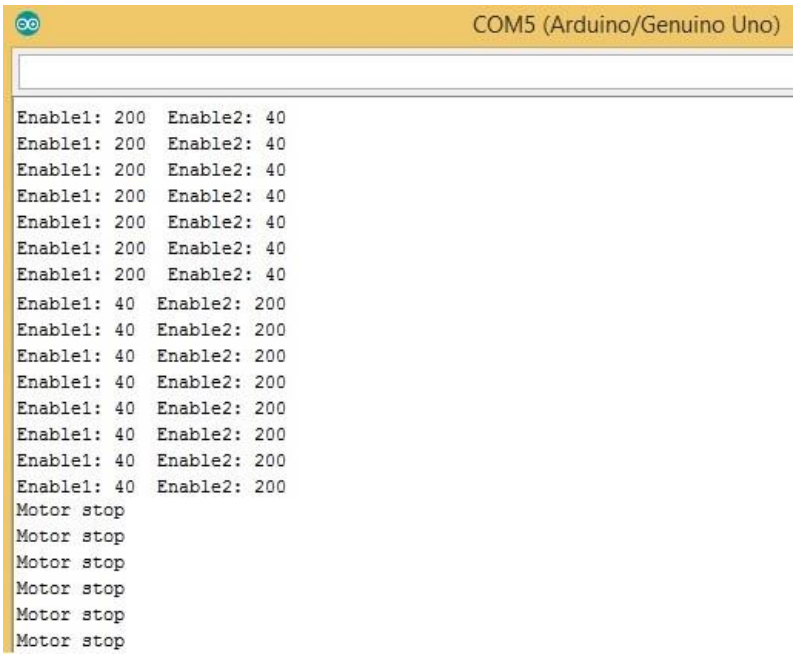
Slika 11: Vezje za preizkus delovanja, primer 2 (vir: avtor naloge).

Programska koda:

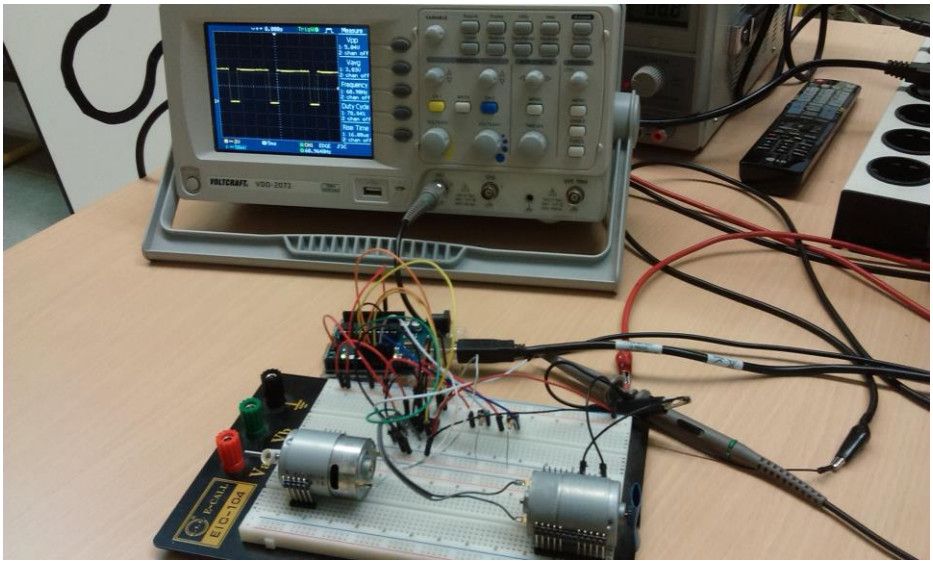
```
/*
 * Preizkus spreminjanja hitrosti vrtenja dveh DC motorčkov z različnima duty cycle PWM signala.
 * Primer => dva DC motorčka krmilimo z dvema tipkama. Če je sklenjena prva tipka, se motorčka
 * vrtita hitreje (duty cycle večji), če je sklenjena druga tipka, se motorčka vrtita počasneje (duty
 * cycle manjši). Frekvenca je konstantna 61 Hz.
 * Timer2 => f=61 Hz
 * mode 3 => Fast PWM
 * preddelitev 1:1024
 * Priključka: OCR2A => pin 11
 *             OCR2B => pin 3
 */
 //Pini za gonilnik L293D:
 #define M_desniA 7    //Signal za desni motorček, priključen na pin 2 na L293.
 #define M_desniB 8    //Signal za desni motorček, priključen na pin 7 na L293.
 #define M_desniE 3    //Enable pin za desni motorček, priključen na pin 1 na L293, PWM.

 #define M_leviA 9    //Signal za levi motorček, priključen na pin 10 na L293.
 #define M_leviB 10    //Signal za levi motorček, priključen na pin 15 na L293.
 #define M_leviE 11    //Enable pin za levi motorček, priključen na pin 9 na L293, PWM.

 int Tipka = 12;
 int Tipka1 = 13;
 int Polozaj_tipke = 0;
 int Polozaj_tipke1 = 0;
```

```
//Spremenljivki za PWM krmiljenje hitrosti vrenja DC motorčka (od 0 do 255):
 int Enable1;
 int Enable2;

void setup()
{
  //Določitev vhodov in izhodov:
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(Tipka, INPUT);
  pinMode(Tipka1, INPUT);
  Serial.begin(9600);

  //Sprememba frekvence PWM signala na Timer2 (mode 3 => FAST PWM, f=61 Hz):
  TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(CS20) | _BV(CS21) | _BV(CS22);                    //Preddelitev 1:1024
}

void loop()
{
  Polozaj_tipke = digitalRead(Tipka);
  Polozaj_tipke1 = digitalRead(Tipka1);

  if((Polozaj_tipke == HIGH) && (Polozaj_tipke1 == LOW))
  {
    OCR2A = 200;
    OCR2B = 40;
    Enable1 = OCR2A;
    Enable2 = OCR2B;

    analogWrite(M_desniE, Enable1);
    digitalWrite(M_desniA, HIGH);          //Desni motor vrtenje naprej, duty cycle 78 %.
    digitalWrite(M_desniB, LOW);

    analogWrite(M_leviE, Enable1);
    digitalWrite(M_leviA, HIGH);           //Levi motor vrtenje naprej, duty cycle 16 %.
    digitalWrite(M_leviB, LOW);

    Serial.print("Enable1: ");
    Serial.print(Enable1);
    Serial.print("  Enable2: ");
    Serial.println(Enable2);
  }

  else
  {
    if((Polozaj_tipke == LOW) && (Polozaj_tipke1 == HIGH))
    {
```

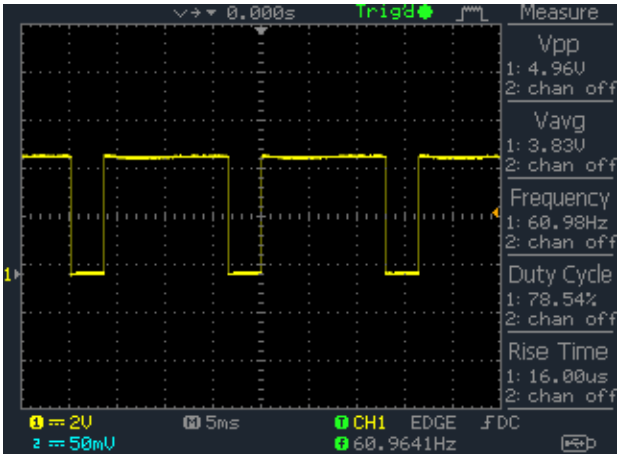
```
    OCR2A = 40;
    OCR2B = 200;

    Enable1 = OCR2A;
    Enable2 = OCR2B;

    analogWrite(M_desniE, Enable1);
    digitalWrite(M_desniA, HIGH);          //Desni motor vrtenje naprej, duty cycle 16 %.
    digitalWrite(M_desniB, LOW);

    analogWrite(M_leviE, Enable1);
    digitalWrite(M_leviA, HIGH);           //Levi motor vrtenje naprej, duty cycle 78 %.
    digitalWrite(M_leviB, LOW);

    Serial.print("Enable1: ");
    Serial.print(Enable1);
    Serial.print("  Enable2: ");
    Serial.println(Enable2);
   }
   else
   {
    digitalWrite(M_desniA, LOW);           //Motor desni stop.
    digitalWrite(M_desniB, LOW);
    digitalWrite(M_leviA, LOW);            //Motor levi stop.
    digitalWrite(M_leviB, LOW);
    Serial.println("Motor stop");
   }
  }
  delay(10);
}
```



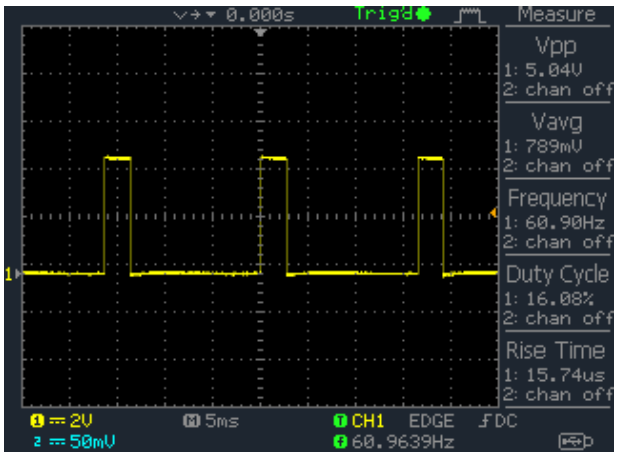
Slika 12: Izpis na serijskem monitorju, (vir: avtor naloge).



Slika 13: Preizkus delovanja z dvema DC motorčkoma in meritve, (vir: avtor naloge).



Slika 14: DC motorček vrti hitreje, OCR2A = 200 (vir: avtor naloge).

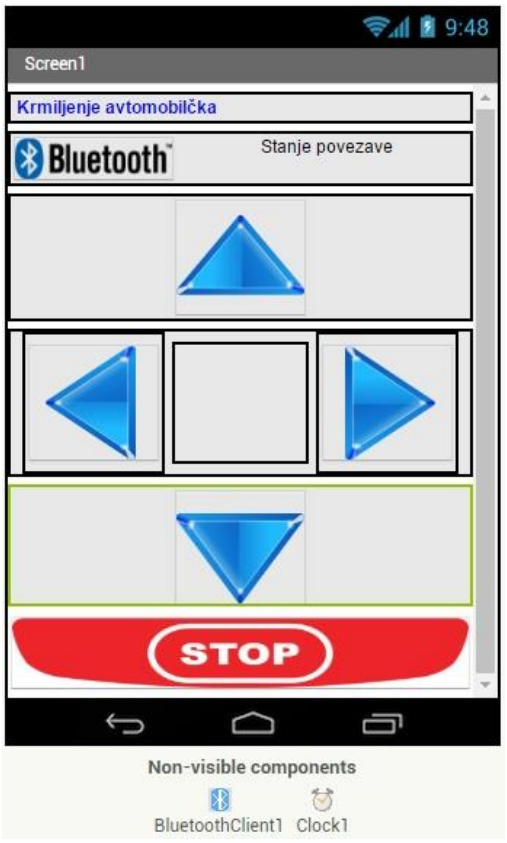


Slika 15: DC motorček vrti počasneje, OCR2A = 40 (vir: avtor naloge).

3.6.3 Izdelava aplikacije za pametni telefon

Aplikacijo (App) za pametni telefon Android smo izdelali v MIT app inventorju, za kar smo morali imeti Googlov račun. Najprej smo projekt poimenovali. Če je ime sestavljeno iz več besed, uporabimo podčrtaje.

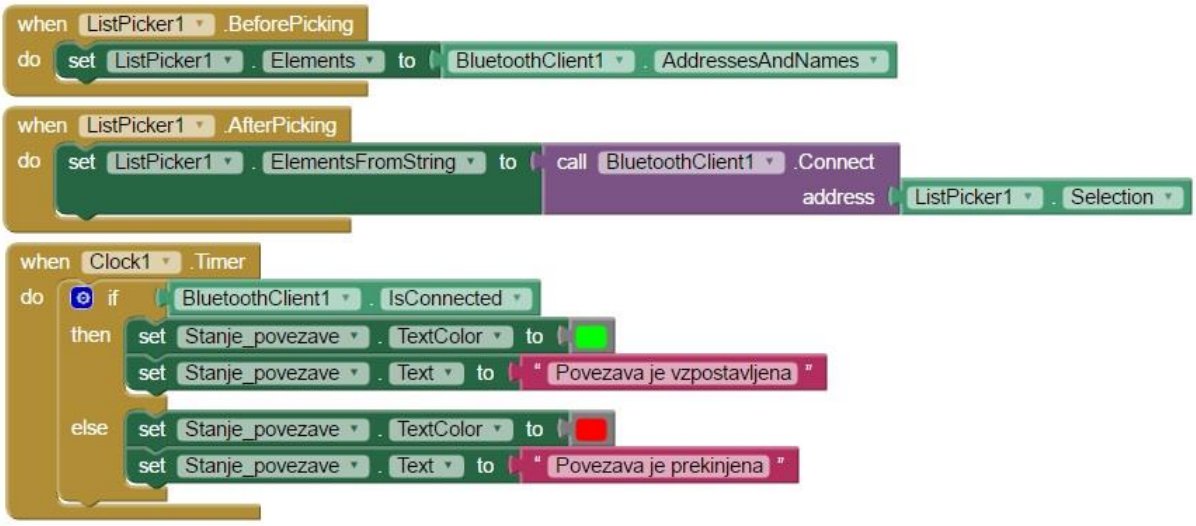
V oknu imamo na razpolago dva gumba, *Designer* in *Blocks*. Začeli smo v oknu *Designer*, kjer imamo na levi strani paleto z orodji (*Palette*). Največje okno je *Viewer*, kjer je prikazan zaslon telefona. Na podlagi razpoložljivih orodij smo oblikovali obliko in položaje posameznih gumbov na zaslonu telefona. Gumbe smo poimenovali. Dodali smo še *BluetoothClient1* in *Clock1*, da pametni telefon zazna Bluetooth napravo.



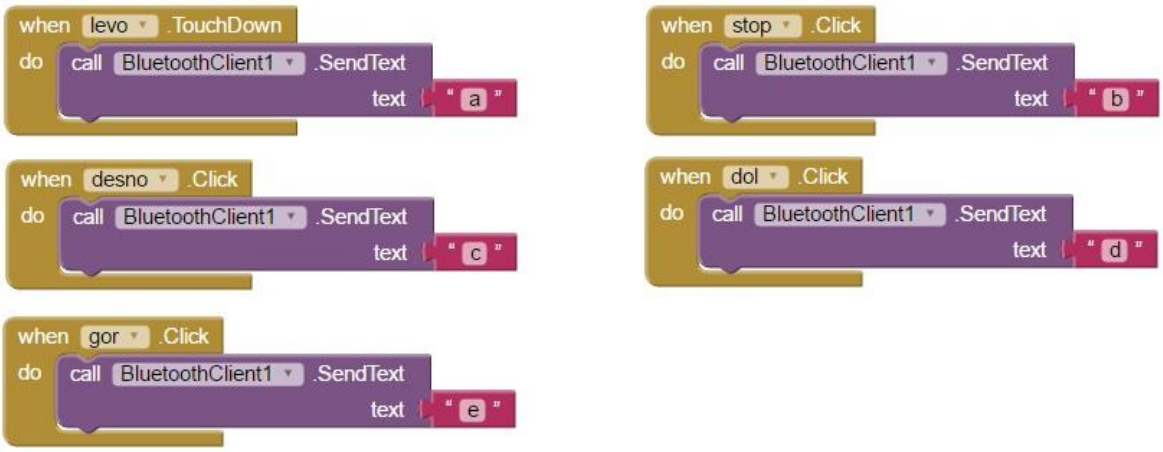
Slika 16: Gumbi na zaslonu pametnega telefona (vir: avtor naloge).

V oknu *Blocks* smo izdelali programsko kodo. Na zaslonu pametnega telefona se nam mora prikazati ali je Bluetooth povezava vzpostavljena ali je prekinjena.

Posameznim gumbom smo določili tekst kode, ki se po pritisku na gumb po Bluetooth povezavi pošlje sprejemnemu modulu. Koda, ki jo modul sprejme, se prikaže na serijskem monitorju okolja Arduino IDE.



Slika 17: Koda za Bluetooth povezavo (vir: avtor naloge).



Slika 18: Koda za gumbe (vir: avtor naloge).

V glavni zanki *viod loop()* neprestano preverjamo, ali je preko serijske komunikacije prišel podatek o pritisnjeni tipki na pametnem telefonu. Če je, se ta tudi izpiše na serijskem monitorju.

```
if (Serial.available() > 0)
  {
    serialA = Serial.read();
    Serial.println(serialA);
  }
```

Ugotovili smo kode, ki se izpišejo ob pritisku na posamezne tipke. Te kode smo uporabili v programu. Uporabili smo stavek switch-case.

Primer kode, če pritisnemo na gumb gor. Avtomobilček vozi naprej:

```
switch (serialA)
   {
   case 6:                 //Koda 6 prejeta, če pritisnemo na gumb gor (avtomobilček naprej).
    OCR2A = 200;           //Pri vrednosti 200 je duty cycle = (200+1)/256 * 100% = 78% => pin 11.
    OCR2B = 200;           //Pri vrednosti 200 je duty cycle = (200+1)/256 * 100% = 78% => pin 3.

    Enable1 = OCR2A;
    Enable2 = OCR2B;

    analogWrite(M_desniE, Enable1);     //Vrednost duty cycle na pin 3 gonilnika L293.
    digitalWrite(M_desniA, HIGH);       //Desni motor vrtenje naprej.
    digitalWrite(M_desniB, LOW);

    analogWrite(M_leviE, OCR2B);        //Vrednost duty cycle na pin 3 gonilnika L293.
    digitalWrite(M_leviA, HIGH);        //Levi motor vrtenje naprej.
    digitalWrite(M_leviB, LOW);

    break;
```

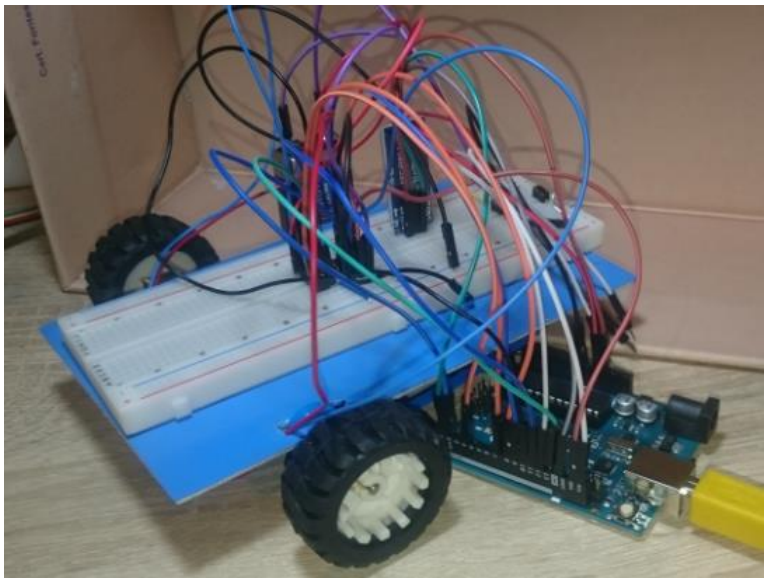
Zavijanje avtomobilčka v levo oziroma v desno smo izvedli tako, da se vrti samo ustrezni motorček. Pri zavijanju v levo smer se vrti samo desni motorček, pri zavijanju v desno smer pa se vrti samo levi motorček. Delovanje smo preizkusili z vezjem, sestavljenim na eksperimentalno ploščico. Dodali smo Bluetooth modul in ga povezali z Arduino Uno. V času zapisovanja programske kode smo morali prekinili RX – TX povezavo med Bluetooth modulom in Arduino Uno, drugače je bilo zapisovanje neuspešno.

3.7 Izdelava makete avtomobilčka

Naprava je delovala je po pričakovanjih. Na izbrano ploščo smo začasno pritrdili dva enosmerna motorčka, ki pa sta bila drugačna od tistih, ki smo jih uporabili pri prejšnjih preizkusih. Vsebujeta prenos vrtenja, tako da se kolesi vrtita počasneje od motorčka.

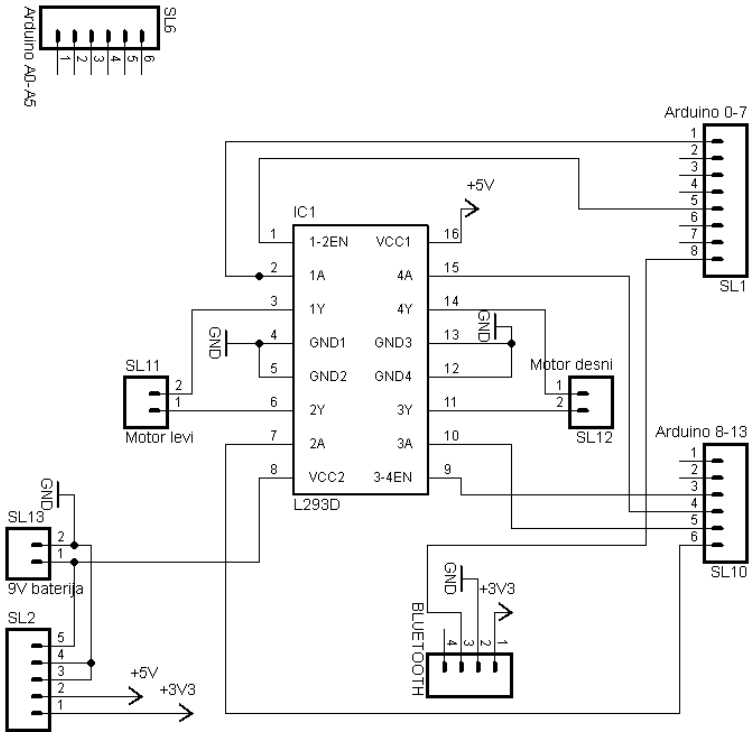
Ko smo z izdelano kodo, zapisano v razvojno ploščo Arduino Uno, preizkusili delovanje, se motorčka s prenosom in pritrjenimi kolesi nista vrtela s hitrostjo, kot sta se v preizkusih uporabljena motorčka. Na motorčkih tudi ni bilo nobenih oznak o tipu motorčka. Nismo poznali podatkov, kakšna mora biti napetost za motorčke in kakšen je prenos. Zato smo eksperimentalno priključili na motorčka enosmerno napetost 9 V in opazovali delovanje. Napetost smo nižali in višali do vrednosti 12 V. Na koncu smo se odločili, da jih krmilimo z napetostjo 9 V, saj bomo za napetostno napajanje razvojne plošče Arduino Uno in hkrati za krmiljenje motorčkov uporabili baterijo 9 V.

Spremeniti smo morali tudi vrednosti za OCR2A in OCR2B da smo dobili primerne vrtljaje motorčkov in s tem primerno hitrost avtomobilčka.

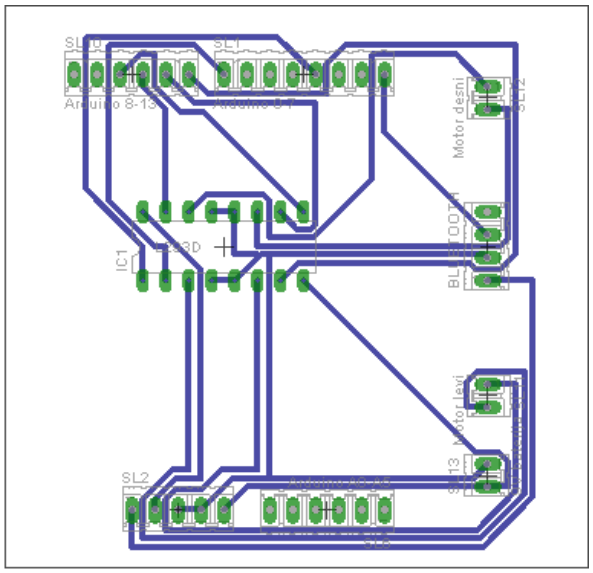


Slika 19: Model za preizkus krmiljenja s pametnim telefonom (vir: avtor naloge).

Za izdelavo končnega modela avtomobilčka smo izdelali tiskano vezje, saj nam je vezje, sestavljeno na eksperimentalni ploščici služilo le za izvajanje preizkusov. V okolju Eagle smo načrtovali elektronsko vezje. Uporabili smo konektorje, ki smo jih v oknu za načrtovanje tiskanine razporedili tako, da je ploščica tiskanega vezja nameščena in priključena na konektorje razvojne plošče Arduino Uno. Za večjo stabilnost tiskanine smo dodali še konektor za analogne vhode, čeprav teh vhodov v nalogi ne potrebujemo.



Slika 20: Elektronski načrt (vir: avtor naloge).



Slika 21: Ploščica tiskanega vezja (vir: avtor naloge).

Ploščico smo izdelali na rezkarju, vstavili in zaspajkali ustrezne elemente in jo vstavili v konektorje razvojne plošče Arduino Uno.

4. REZULTATI

Avtomobilček kot končni izdelek raziskovalne naloge vozi po ukazih, ki jih prejme iz pametnega telefona. Aplikacija za pametni telefon je izdelana. Komunikacija od pametnega telefona do avtomobilčka po Bluetooth povezavi deluje. Spoznali smo tudi vlogo časovnikov v mikrokontrolerju, ter enega od njih uporabili za spremembo frekvence PWM signala, z ustreznima registroma tega časovnika pa smo spreminjali duty cycle tega signala. Naša predvidevanja, ki smo jih izračunali, so se izkazala kot pravilna saj so nam meritve z osciloskopom to potrdile. Ocenjujemo, da je cilj, ki smo si ga zadali, izpolnjen.

5. DRUŽBENA ODGOVORNOST

Daljinsko upravljanje vozila je lahko zelo koristno pri reševanju ljudi in živali ob naravnih nesrečah. Z njim lahko dostopamo na težko dostopna, pa tudi nevarna mesta. Z ustreznimi senzorji, ki jih lahko takšno vozilo vsebuje, lahko kontroliramo prisotnost morebitnih poškodovanih ljudi in živali na nevarnih mestih, pa tudi morebitne strupe in druge nevarnosti. S takšnimi podatki bi lahko reševalci ustrezno ukrepali, preden se sami pojavijo na izpostavljenih območjih.

Tiskano vezje smo izdelali s pomočjo rezkarja. S tem smo pripomogli k varovanju okolja, saj nismo uporabljali določenih kemikalij, ki so potrebne pri jedkanju tiskanega vezja.

6. VIRI

http://www.dx.com/p/bluetooth-board-module-4-pin-121326#.VLPAxXus2ZR (28. 09. 2016)

http://www.nanoelektronika.si/motor-12vdc-19w-mab800-11417/ (28. 09. 2016)

http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=l293&fileType=pdf (15. 10. 2016)

https://sites.google.com/site/solaelektronikesers/home (10. 10. 2016)