Izrada makete daljinski upravljanog automobila

Markutović, Hrvoje

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

Sveučilišni studij

IZRADA MAKETE DALJINSKI UPRAVLJANOG AUTOMOBILA

Završni rad

Hrvoje Markutović

Osijek, 2016

Obrazac Z1 – Obrazac za ocjenu završnog rada

Izjava o originalnosti završnog rada

SADRŽAJ

1. UVOD

Cilj ovog završnog rada je izraditi maketu daljinski upravljanog automobila s ugrađenim mikroupravljačkim sustavima. Sustav može biti podijeljen na dvije cjeline: daljinski upravljač i maketa automobila. Potrebno je omogućiti daljinsko upravljanje četiri istosmjerna motora pomoću pulsno – širinske modulacije i Arduino RF modula za bežičnu komunikaciju. Daljinski upravljač imat će odašiljač i joystick (analog) čiji će položaj biti obrađen te poslan maketi automobila pomoću Arduino modula. Nakon uspješnog slanja, informacija će biti obrađena i poslana na četiri istosmjerna motora. Maketa automobila će imati mogućnost skretanja okretanjem kotača u međusobno suprotnim smjerovima.

1.1. Zadatak završnog rada

Zadatak završnog rada je omogućiti daljinsko upravljanje makete automobila. U radu treba izraditi daljinski upravljač i maketu automobila te programirati mikroupravljače kako bi se dobilo željeno ponašanje sustava.

2. PRIMJENJENE TEHNOLOGIJE I ALATI

2.1. Arduino razvojno okruženje

Najprije je potrebno odabrati platformu pomoću koje će se realizirati sustav. U ovom sklopu koristiti će se Arduino računalna i softverska platforma te moduli potrebni za realizaciju sustava koji su prikazani u tablici 2.1.

Tablica 2.1. Razvojni sustavi te moduli

Naziv	**Mikrokontroler**	**Model**	**Standard**	**Dimenzija**	**Cijena**
Arduino UNO	ATmega328P	-	-	53.3mm x 50.8mm	30 HRK
Arduino NANO	ATmega328P	-	-	17.78mm x 43.18mm	20 HRK
Transmitter	ATmega328P	MX-FS-03V	433.92MHz	16mm x 11mm	4 HRK
Receiver	ATmega328P	MX-05V	433.92MHz	10.5mm x 43.5mm	4 HRK
Joystick	-	KY-023	-	32mm x 24mm	10 HRK
H-bridge	L9110S	*Motor driver*	-	29.2mm x 23mm	7 HRK

Navedena platforma je odabrana jer je to otvorena računalna i softverska platforma koja nam omogućava stvaranje uređaja i komponenti [1], te omogućava da spojimo računalo sa vanjskim svijetom. Rabi 8-bitne Amtel AVR mikrokontrolere te je zato jednostavna, mala i jeftina platforma.

Pisanje koda i komunikaciju Arduino sklopovlja s računalom omogućava *Arduino Software* (IDE) – eng. *Arduino Integrated Development Environment.* Arduino biblioteke za module su otvorenog izvornog koda i omogućava dodavanje novih funkcionalnosti i samostalno ispravljanje pogrešaka.

2.2. Arduino Uno razvojna maketa

Arduino Uno pruža set analognih ulaza, što je jako bitno pri odabiru same platforme i modula potrebnih za realizaciju daljinskog upravljača [2]. To je mikrokontroler baziran na „ATmega328P“ čipu. Ima 14 digitalnih *input/output* pinova, šest analognih ulaza, 16 MHz kvarcni kristal, USB vezu,*ICSP header* i *reset* tipku. Tehničke specifikacije moguće je vidjeti na slici 2.1.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Slika 2.1. Tehničke karakteristike Arduino Uno mikroupravljača [2]

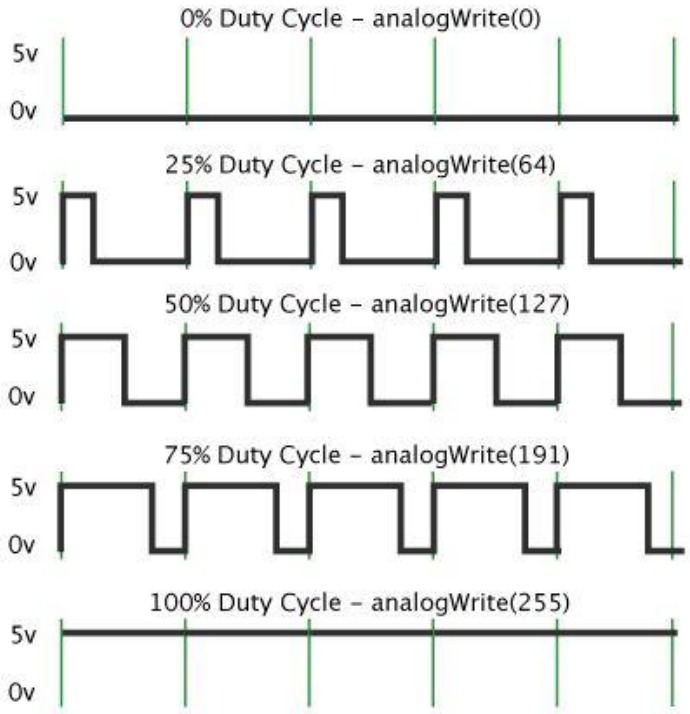
Slika 2.2. Arduino Uno [2]

2.3. Arduino Nano razvojna maketa

Arduino Nano ima set analognih izlaza, odnosno PWM (eng. *pulse-widthmodulation* – pulsno širinska modulacija) gdje se vrlo jednostavno može dobiti promjenjiva brzina električnih motora, a time i brzina kretanja makete. PWM (pulsno - širinska modulacija) [6] je tehnika kojom dobijemo analogni rezultat sa digitalnim značenjem, odnosno koristi se digitalna kontrola za stvaranje kvadratnog vala. Nizom električnih impulsa nastaje električni signal. Vremenskim trajanjem samog električnog impulsa i pauzom između istog dobivamo željeni rezultat. U slučaju ovoga rada to je brzina kretanja motora. Jedan od primjera primjene PWM je podešavanje svjetline LED žarulje. Arduino Nano je malih dimenzija, lako se može postaviti na eksperimentalnu pločicu, koristi „ATmega168“ čip, spaja se putem Mini B – USB kabla. Tehničke specifikacije Arduino Nano su vidljive na slici 2.3.

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"
Length	45 mm
Width	18 mm
Weigth	5 g

Slika 2.3. Tehničke karakteristike Arduino Nano mikroupravljača [3]



Slika 2.4. PWM modulacija [5]

Za pogon makete automobila odabrano je četiri istosmjerna (DC – eng. *directcurrent*) motora čije se karakteristike mogu vidjeti u tablici **2.2.**

Tablica 2.2. Karakteristike istosmjernog motora

Napon	DC 6V
Struja	400 mA
Omjer	48:1
Broj okretaja u minuti (sa kotačem)	240
Radijus kotača	66 mm
Brzina automobila (metara u minuti)	48
Masa motora (g)	50
Buka	<65 dB

2.4. Upravljanje DC motorom pomoću H – mosta

Problem koji se može javiti u primjeni istosmjernog motora jeste vrlo nizak okretni moment na niskim okretajima. Kada je motor u stanju mirovanja ima malu sposobnost pokretanja nekog

predmeta. To je razlog odabira istosmjernog motora sa definiranim prijenosom i kotačima. Kako bi postojala mogućnost promjene brzine vrtnje motora odabrana je tehnologija pulsnoširinske modulacije (PWM). Kako bi bila moguća primjena te tehnologije, te postoji potreba za vrtnjom motora u oba smjera može biti primijenjen *H-bridge* modul za Arduino.

Razlog za odabir *H-bridge* modula je to što omogućava promjenu polariteta na izlazima te tako omogućava na jednostavan način kontrolu vrtnje motora u oba smjera. Potrebno je primijeniti elektronički spoj koji sadrži tranzistore. Ako se koriste samo dva tranzistora, moguće je pokretanje motora samo u jednom smjeru. Za mogućnost promjene smjera vrtnje potrebno je koristiti četiri tranzistora. Odabrani modul ima sposobnost kontroliranja dva istosmjerna motora te se treba koristiti dva modula.

Za postizanje potrebnog napona potrebno je koristiti regulator napona. Najbolji izbor za maketu automobila je L7806CV regulator napona na 6V koji daje maksimalan izlaz struje od 1500 mA. Na maketi automobila će biti korišteno dva regulatora napona koji će regulirati napon na svaki od *H-bridge* modula.

2.5. RF primopredajnik i VirtualWire biblioteka

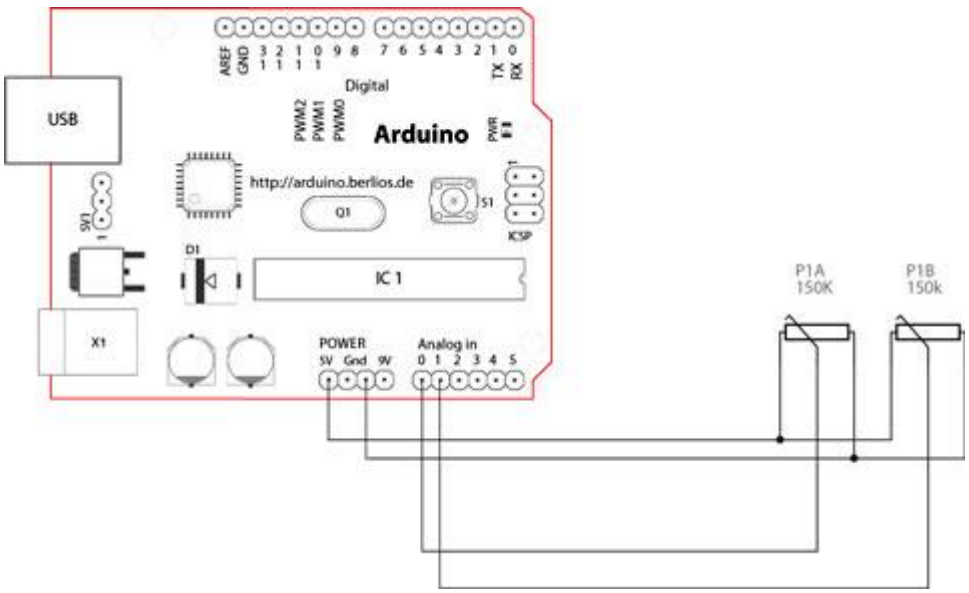
Za uspješno slanje informacija koje trebaju biti očitane sa *Joystick* modula na daljinskom upravljaču te za definiranje varijabli potrebno je koristiti biblioteku *VirtualWire* koja pruža značajke slanja kratkih poruka bez potrebe adresiranja, retransmisije i potvrde, primjenom amplitudne modulacije. Podržava jeftine i pristupačne odašiljače i prijemnike. Isto tako, na prijemnoj strani (maketa automobila) je potrebno primijeniti *VirtualWire* biblioteku za primanje informacije kako bi mogla biti dalje obrađena te poslana pogonskom sustavu makete automobila.

Kriterij za odabir *Transmitter/Receiver* modula jest njegova jednostavnost i fleksibilnost. Više je načina pisanja koda za traženi rezultat. Prednost su njegove dimenzije koje su zanemarive te cijena. Odašiljač odašilje signal do 90 m na otvorenom području. Radna frekvencija odašiljača je 433MHz i snage 25mW. Brzina prijenosa podataka je do 10Kbps. Prijemnik radi na istoj frekvenciji.

2.6. Analogno digitalni modul Joystick-a

Joystick je ulazni modul za Arduino [4] koji daje X i Y analogna očitanja smjera i digitalno čitanje tipkala. Modul se sastoji od dva potenciometra. Očitanja su analogna (vrijednosti od 0-1023) i može se precizno odrediti položaj potenciometra. Potenciometri stvaraju analogne vrijednosti,

otpor se smanjuje ili povećava u ovisnosti o položaju poluge. Izlazni napon ovisi o otporu koji daju potenciometri.



Slika 2.5. Shematski prikaz spoja *Joystick* modula [4]

Slika 2.6. *Joystick* modul [4]

3. REALIZACIJA SUSTAVA

3.1. Montažne i elektroničke sheme sustava

Sustav se sastoji od dva glavna dijela: daljinski upravljač i maketa automobila. Sustav je dizajniran tako da se može mijenjati brzina kretanja makete po želji. Upravljač se sastoji od Arduino Uno pločice, Arduino *Joystick* modula i *Transmitter* modula za Arduino.

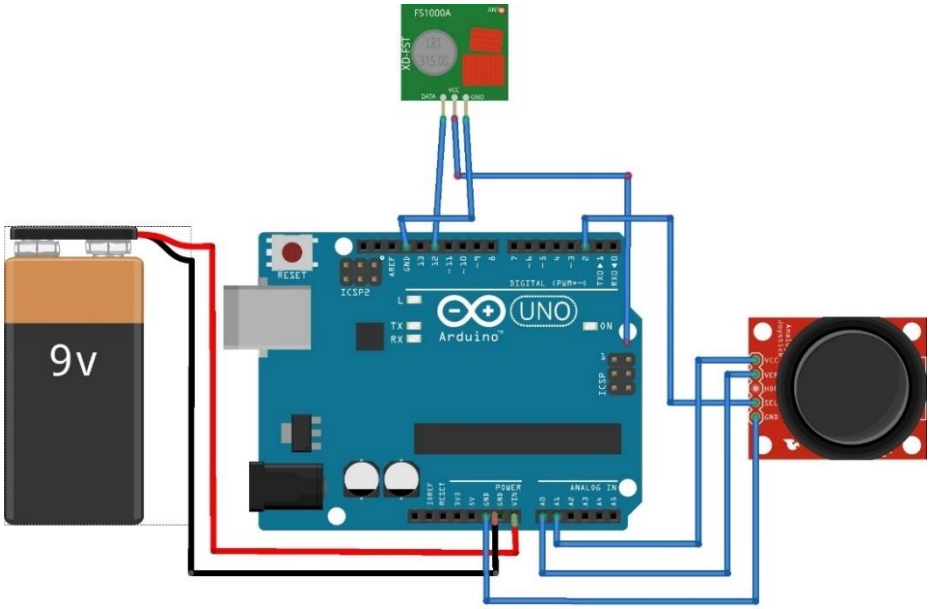
Joystick modul za arduino omogućava kretanje poluge u svim smjerovima te je vrlo jednostavno zadati smjer i brzinu kretanja makete. Zahvaljujući potenciometru dobivena su analogna očitanja [4]. Vrijednosti koje su dobivene moraju biti dalje obrađene. Podatke (brojeve) koji su očitani trebaju biti spremljeni u niz. Arduino koristi C++ programski jezik te je kod potrebno napisati u tom jeziku. Za spremanje niza podataka koriste se jednodimenzionalna polja. Definirani smjerovi kretanja po „koordinatama“ prikazani su na slici 3.1.

Vrijednost je potrebno pretvoriti u tip podatka „char“, te poslati. Za slanje je potrebno uključiti biblioteku *VirtualWire* jer se za slanje podataka spojen *RF* modul za Arduino.Za primanje podataka koristi se *RF* modul, prijemnik koji se nalazi na maketi automobila te je spojen s Arduino Nano pločicom. Primljene podatke treba pretvoriti u podatak tipa „integer“ i zatim u interval od nula do 255. Zadani interval definira brzinu kretanja motora. Nakon pozivanja funkcije *analogWrite(),* digitalni izlaz na Arduino pločici generira kvadratni signal (PWM modulacija).

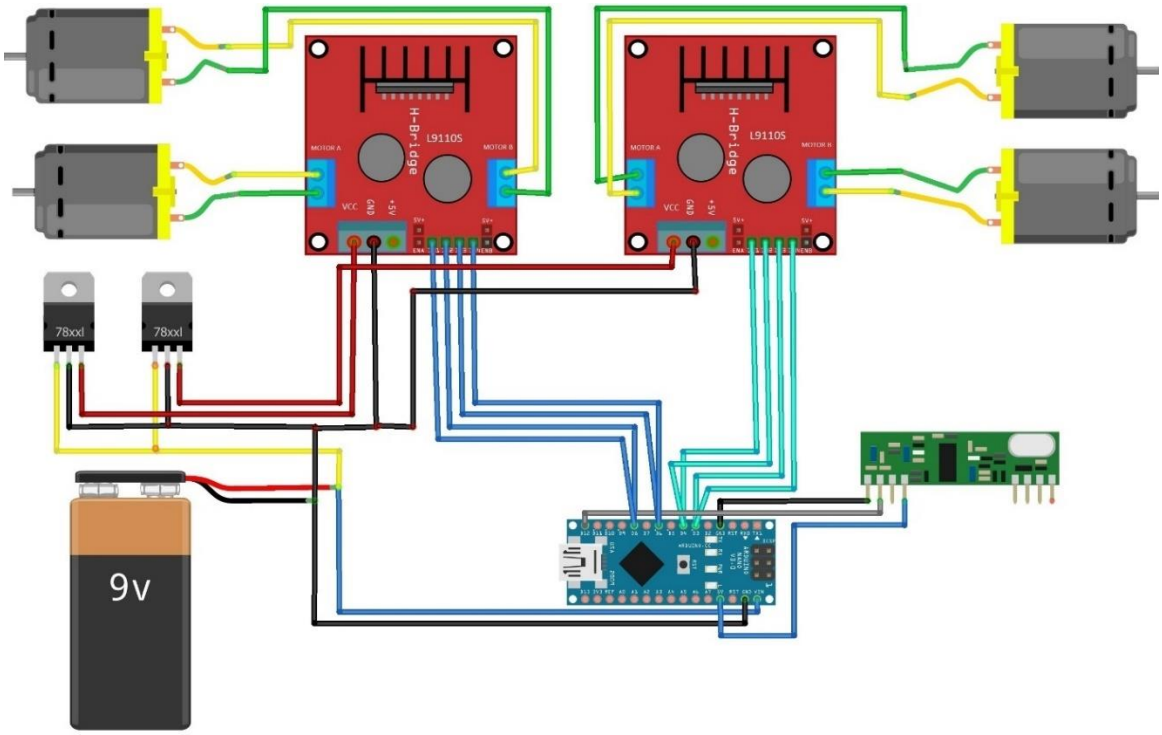
Kako bi se omogućilo kretanje motora, koristi se H-bridge modul koji omogućava da se napajanje motorima dovede izravno od izvora napajanja, a signal koji se generira na izlazima na Arduinu definira smjer i brzinu kretanja motora. Generiranje signala na pojedinom izlazu definirano je u kodu (pogledati prilog B). Prilikom stajanja ili jako niskih okretaja, motor ima jako mali okretni moment. Kako bi problem okretnog momenta bio riješen, koriste se prijenosi (eng. *gear*) na izlazu mehaničkog gibanja motora koji su direktno spojeni na kotače. U maketi automobila se koristi četiri istosmjerna motora, te je za odluku o korištenju izvora energije potrebno izmjeriti minimalne zahtjeve sustava. Maketa je spojena na laboratorijski izvor napajanja, na kojoj je namješten izlaz 9V. Za normalan rad makete potrebno je 1.68A struje pri 9V. Najbolji izbor izvora energije je litij ionska 9V baterija.

Prvi korak u izradi makete je izrada elektroničke sheme. Za izradu sheme korišten je „Fritzing“ *software, opensource* inicijativa za razvoj amaterskog ili CAD softvera za dizajn hardvera elektronike. Treba napraviti dvije odvojene sheme, shema daljinskog upravljača i shema makete

automobila. Na slici 3.2. je prikazana shema daljinskog upravljača. Vrlo je jednostavna jer se koristi tek nekoliko elemenata. Kao izvor energije koristi se baterija napona 9V, *Transmitter*, *Joystick* modul i Arduino pločica na koju su spojeni svi dijelovi.

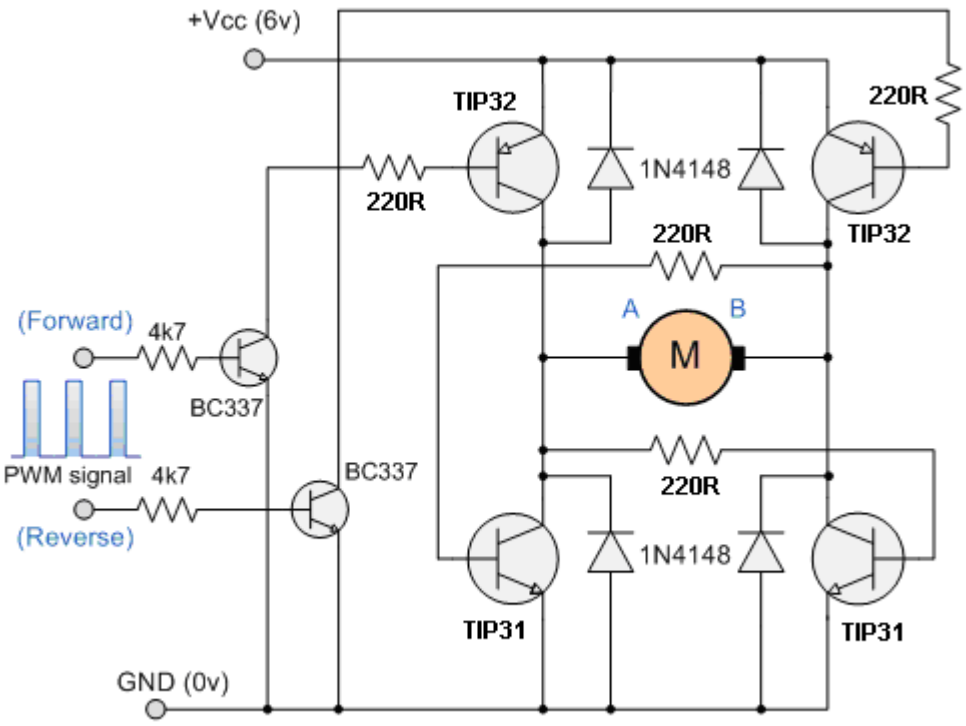


Slika 3.1. Montažna shema daljinskog upravljača



Slika 3.2. Montažna shema makete automobila

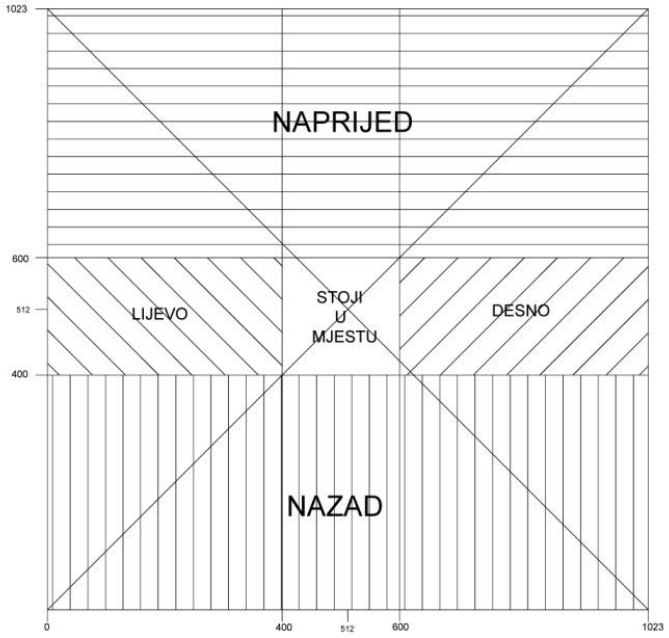
Na slici 3.3 prikazana je shema makete automobila. Kao izvor napajanja korištena se 9V baterija. Radi boljih performansi stavljena je Li-ion baterija. Na izvor napajanja je spojeno dva regulatora napona na 6V, na koje su spojena dva *H-bridge driver-a*, jedan za dva motora. Arduino pločica se napaja direktno sa izvora napajanja. Dva digitalna (PWM) izlaza sa Arduino pločice su spojena na svaki *H-bridge* driver. Preostala dva ne moraju biti PWM izlazi (definiraju smjer).



Slika 3.3. Elektronička shema H-mosta. Slika preuzeta iz [8]

3.2. Programsko rješenje sustava

Za implementaciju koda korišten je *Arduino Software* (IDE) – eng. *Arduino Integrated Development Environment.* Program je napisan u C++ programskom jeziku. Za realizaciju daljinskog upravljača primijenjen je razvojni sustav Arduino Uno. Kako bi slanje informacija putem *RF* modula bilo moguće, uključena je biblioteka *VirtualWire.* Najprije je definiran izlaz na Arduinu koji će generirati signal i spojen je sa odašiljačem. Svaki Arduino program treba definirati najmanje dvije funkcije – *setup* i *loop*. *Setup* funkcija služi za inicijalizaciju sklopova i izvršava se samo jednom, nakon pokretanja mikrokontrolera. *Loop* funkcija poziva se u beskonačnoj petlji. Funkcije su definirane u datoteci *Transmitter.ino* (prilog A). Funkcije u *setup* funkciji su funkcije konfiguracije, definira izlazni pin te brzinu slanja podataka. U *loop* funkciji je definirano ponašanje sklopa. Analognim očitanjem položaja poluge na *Joysticku* dobivene su vrijednosti u intervalu od nula do 1023 na X osi i isto tako na Y osi. Za slanje, podaci trebaju biti poslagani u niz te je za to korišteno jednodimenzionalno polje. Podaci su pretvoreni u tip podatka „*char*“ zbrajanjem vrijednosti brojem 48. U *for* petlji višeznamenkasti broj se odvaja i pojedino sprema u niz (polje). Pomoću funkcije za slanje „*vw_send(message, length)*“ koja je definirana u biblioteci *VirtualWire* šalju se podaci. „*Message*“ predstavlja niz bitova za slanje dok „*lenght*“ predstavlja broj bajtova pohranjenih u nizu. Funkcija „*vw_wait_tx();*“ definira čekanje dok cijela poruka ne bude poslana. Cijeli kod nalazi se u prilogu (prilog A).



Slika 3.3. Definirani smjer kretanja makete ovisno o položaju poluge *Joystick* modula

Na prijemnoj strani koristi se razvojni sustav Arduino Nano. Kao i za daljinski upravljač, mora biti uključena biblioteka *VirtualWire*. Najprije je definiran *pin* koji je spojen sa prijemnikom koji će primati podatke. Definirani su i ostali izlazi koji će slati informacije o položaju poluge na istosmjerne motore. Nazivi varijabli odgovaraju nazivima ulaza na *H-bridge* modulu radi lakšeg snalaženja. U *setup* funkciji je definiran ulaz te brzina prenošenja informacije. U *loop* funkciji se događa obrnuti slijed od slijeda kod daljinkog upravljača. *Char* tip podatka je pretvoren u tip podatka *integer* oduzimanjem primljenog podatka brojem 48, potom u interval brojeva od 0 do 1023. Definirani su uvjeti kretanja makete ovisno o primljenim podacim varijable x i y. Navedeni interval je potrebno pretvoriti u interval od 0 do 255 jer je funkcija *analogWrite()*; definirana u tom intervalu [5]. Ovisno o vrijednosti varijabli definirana je brzina (interval od nula do 255) i smjer (*if* petlja) kretanja. Ovisno o položaju poluge, poziva se određena funkcija koja je definirana nakon funkcije *loop*. Kod se može vidjeti u prilogu (Prilog B). Na *pwm pin (*spojen na istosmjerni motor) je potrebno definirati vrijednosti preuzete iz *if* petlje (brzina okretanja motora). Drugi *pin* nije potrebno da bude *pwm* jer definira samo smjer okretanja motora.

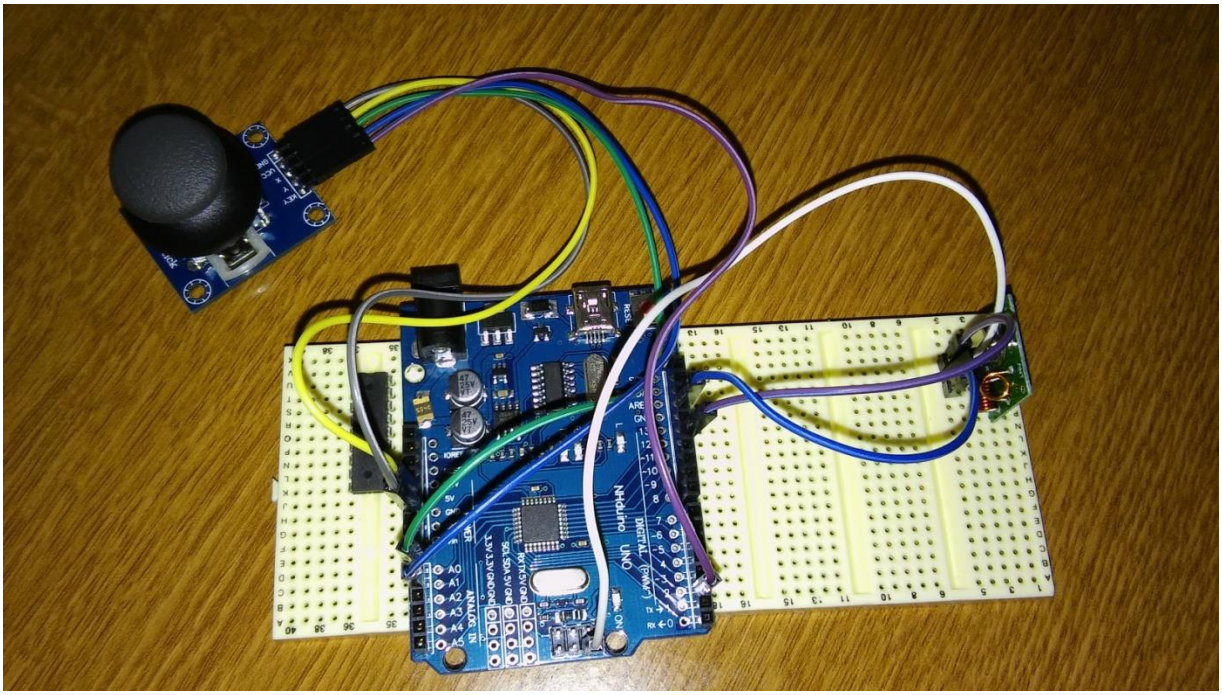
Ponašanje motora ovisno o dovedenom signalu može se vidjeti u tablici 3.1.

Tablica 3.1. Ovisnost ponašanja istosmjernog motora o ulaznom signalu

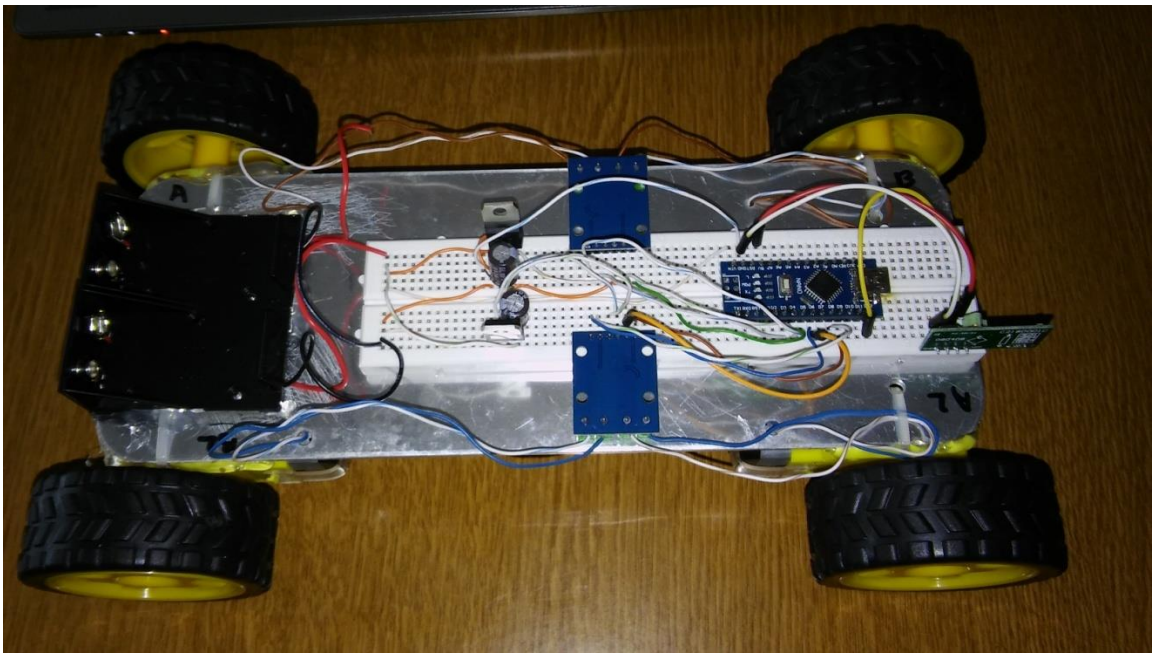
A - A	A - B	MOTOR
LOW	*LOW*	*STOP*
HIGH	*LOW*	*FORWARD*
LOW	*HIGH*	*REVERSE*
HIGH	*HIGH*	*STOP*

3.3. Izgled gotovog sustava

Kao platforma na koju se slažu komponente odabran je lim debljine 3 mm, te su probušene po dvije rupe za jedan motor. Jedna rupica služi kako bi se provukle žice koje idu od motora do *H-bridge*-a, a kroz drugu je provučena plastična vezica koja sprječava motorima da se otrgnu. Za lakše spajanje svih komponenata korištena je eksperimentalna pločica. Pomični dijelovi su učvršćeni topljenom plastikom. Na Arduino Uno (korišten u daljinskom upravljaču) je moguće spojiti sve module izravno.



Fotografija 3.1. Daljinski upravljač



Fotografija 3.2. Maketa automobila

Najbolji odabir napajanja jest Li-ion eksterna baterija kapaciteta 2000 mAh i napona 9V jer omogućava rad četiri istosmjerna motora oko pola sata neprekidnog rada uz relativno brzo punjenje od približno tri sata. Maksimalna brzina kretanja makete je oko 50 metara u minuti i moguća je promjena broja okretaja motora od nula do 240 u minuti pomoću daljinskog upravljača. Udaljenost na kojoj je moguće upravljati maketom je oko 80 metara na otvorenom prostoru.

4. ZAKLJUČAK

U ovom radu izrađena je maketa daljinski upravljanog automobila s ugrađenim mikroupravljačkim sustavima. Sustav se sastoji od dva dijela: daljinski upravljač i maketa automobila. Sustav je uspješno sastavljen te radi kako je zamišljeno. Prednost korištenja ove tehnologije je to što omogućava prilagođavanje brzine kretanja makete (zahvaljujući primjeni pulsno – širinske modulacije i Arduino *H-bridge* modula). Minimalni zahtjevi izvora energije su minimalno 1.7 A te 9V zbog četiri istosmjerna motora koji zahtijevaju veliku snagu te troše puno energije. Dulji rad sustava bio bi moguć korištenjem baterije većeg kapaciteta i veće snage (najbolji izbor je 9V litij ionska baterija) te programskim ograničenjem brzine okretanja istosmjernih motora.

LITERATURA

[1] Razvojno okruženje Arduino, https://www.arduino.cc/en/Guide/HomePage, ožujak 2016.

[2] Arduino Uno, https://www.arduino.cc/en/Main/ArduinoBoardUno, travanj 2016.

[3] Arduino Nano, https://www.arduino.cc/en/Main/ArduinoBoardNano, travanj 2016.

[4] Joystick modul Arduino, https://www.arduino.cc/en/Tutorial/JoyStick, lipanj 2016.

[5] Arduino PWM modulacija, https://www.arduino.cc/en/Tutorial/PWM, lipanj 2016.

[6] PWM modulacija, http://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/, travanj 2016.

[7] Podaci o L9110 čipu, http://www.elecrow.com/download/datasheet-l9110.pdf, lipanj 2016.

[8] H-most elektronička shema, http://www.talkingelectronics.com/projects/H-Bridge/images/H-Bridge-6.gif, lipanj 2016.

SAŽETAK

Naslov: Izrada makete daljinski upravljanog automobila

U radu je napravljena maketa daljinski upravljanog automobila i daljinski upravljač. Sustav se sastoji od dvije cjeline: maketa automobila i daljinski upravljač. U radu su razmotreni alati i tehnologije kojima je moguće riješiti probleme i sastaviti maketu. Mikroupravljački sustavi su ugrađeni u maketu automobila i daljinski upravljač. Nakon opisa primijenjenih alata i tehnologija, pojašnjena je realizacija sustava: izrada makete, izrada daljinskog upravljača te pisanje koda nužnog za definiranje ponašanja sklopa. Sustav je uspješno realiziran.

Ključne riječi: Arduino, PWM modulacija, mikroupravljač, C++, elektronika

ABSTRACT

Title: Development of remote controlled car model

This paper aims to elaborate on a breadboard of a remote control car and a remote controller. The system consists of two parts: a car breadboard and a remote controller. The paper analyzes tools and technologies used to solve problems and create a breadboard. Microcontrolling systems are installed in the car breadboard and the remote controller. Upon describing the applied tools and technologies, the procedures of creating the breadboard, remote controller and writing a code required for the circuit are explained. The system is successfully applied.

Keywords: Arduino, PWM modulation, microcontroller, C++, electronics

ŽIVOTOPIS

Hrvoje Markutović rođen je 26. listopada 1994. godine u Rijeci, Republika Hrvatska. Živi u mjestu Tovarnik. Godine 2009. završava osnovnoškolsko obrazovanje u Osnovnoj školi „Antun Gustav Matoš“ u Tovarniku, te Glazbenu školu u Vinkovcima. Iste godine upisuje Gimnaziju Matije Antuna Reljkovića u Vinkovcima koju završava 2013. godine. Nakon srednje škole svoje obrazovanje nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku koji upisuje 2013. godine. Trenutno je student III. godine preddiplomskog studija računarstva.

PRILOG A. Program za Arduino Uno razvojni sustav (*Transmitter*)

```
#include <VirtualWire.h>
const int trans_pin = 12;
void setup()
{
  pinMode(trans_pin,OUTPUT);
  vw_set_tx_pin(trans_pin);
  pinMode(5,INPUT_PULLUP);
  Serial.begin(9600);
  vw_setup(2000);       // Bits per sec
  vw_set_ptt_inverted(true);
}

void loop()
{
  int x = analogRead(A0);          // 16-bit, 2-Byte
  int y = analogRead(A1);          // 16-bit, 2-Byte
  uint8_t key = 1-digitalRead(5); //  8-bit, 1-Byte

  Serial.print(x);
  Serial.print(" ");
  Serial.print(y);
  Serial.print(" ");
  Serial.println(key);

  char data[VW_MAX_MESSAGE_LEN];
  int p = 1;
  for(int i=0; i<4; i++){
    data[3-i] = (x/p)%10 + 48;
    data[7-i] = (y/p)%10 + 48;
    p *= 10;
  }
  data[8] = key+48;
  data[9] = 0;

  Serial.println(data);

  vw_send((uint8_t *)data, 9);
  vw_wait_tx();

}
```

PRILOG B. Program za Arduino Nano razvojni sustav (*Receiver*)

```
#include <VirtualWire.h>

const int AIA1 =  3;
const int AIB1 =  4;
const int AIA2 =  6;
const int AIB2 = 8;
const int rec_pin = 12;

void setup()
{
    pinMode(AIA1, OUTPUT);
    pinMode(AIB1, OUTPUT);
    pinMode(AIA2, OUTPUT);
    pinMode(AIB2, OUTPUT);

    pinMode(12, INPUT);

    Serial.begin(9600);
    //Serial.println("setup");
    vw_set_rx_pin(rec_pin);

    vw_setup(2000);  // Bits per sec
     vw_rx_start();
}

void loop()
{
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;

  if (vw_get_message(buf, &buflen)) // Non-blocking
  {

    int x,y,key;

    x = int(buf[7]-48) + int(buf[6]-48)*10 + int(buf[5]-48)*100 + int(buf[4]-
48)*1000;
    y = int(buf[3]-48) + int(buf[2]-48)*10 + int(buf[1]-48)*100 + int(buf[0]-
48)*1000;
    key = int(buf[8]-48);

    Serial.print(x);
    Serial.print(" ");
    Serial.print(y);
    Serial.print(" ");
    Serial.println(key);
int brzina;
    if(y>600){
    brzina=byte(float(255)*(float(y)-float(600))/float(1023-600));
    forward(byte(brzina));
    }
```

```
    else if(y<400){
    brzina=byte(float(255)*(float(1)-(float(y)/float(400))));
    backward(byte (brzina));
    }
     else if(x<400){
      brzina=byte(float(255)*(float(1)-(float(x)/float(400))));
      left(byte(brzina));
    }

    else if(x>600){
      brzina=byte(float(255)*(float(x)-float(600))/float(1023-600));
      right(byte(brzina));
    }

    else if((400<x<600)&&(400<y<600)){
      stani();
    }
   }
}

void stani()
{
    digitalWrite(AIA1, LOW);
    digitalWrite(AIB1, LOW);
    digitalWrite(AIA2, LOW);
    digitalWrite(AIB2, LOW);
    }
void forward(byte brzina)
    {
     analogWrite(AIA1, brzina);
     analogWrite(AIB1, 0);
     analogWrite(AIA2,float(255)- brzina);
     analogWrite(AIB2, 255);
    }

   void backward(byte brzina)
    {
    analogWrite(AIA1, float(255)-brzina);
    analogWrite(AIB1, 255);
    analogWrite(AIA2, brzina);
    analogWrite(AIB2, 0);
  }


void left(byte brzina)
{
  analogWrite(AIA1, float(255)-brzina);
  analogWrite(AIB1, 255);
  analogWrite(AIA2, float(255)-brzina);
  analogWrite(AIB2, 255);
}

void right(byte brzina)
```

```
{
  analogWrite(AIA1, brzina);
  analogWrite(AIB1, 0);
  analogWrite(AIA2, brzina);
  analogWrite(AIB2, 0);

}
```