Digitalna vaga za mjerenje statičkog elisnog potiska

Šimara, Eugen

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

DIGITALNA VAGA ZA MJERENJE STATIČKOG ELISNOG POTISKA

Završni rad

Eugen Šimara

Osijek, 2017

SADRŽAJ

1.UVOD

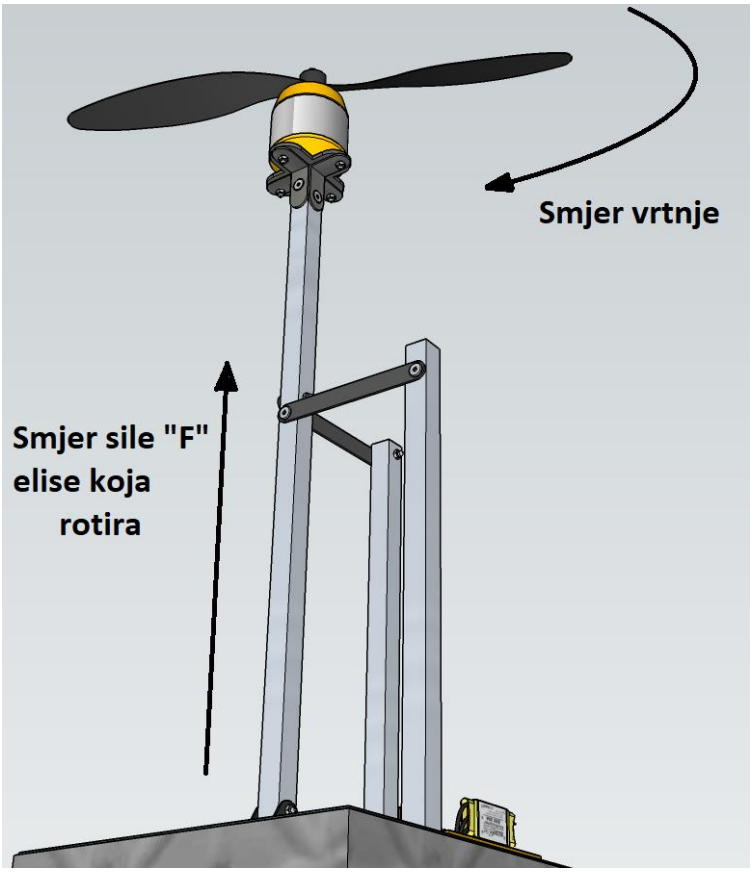
Ovim radom detaljnije ćemo se upoznati s "Digitalnom vagom za mjerenje statičkog elisnog potiska". Kako i njeno samo ime kaže radi se digitalnoj vagi koja služi za mjerenje elisnog potiska različitih vrsta elisa i trofaznih motora bez četkica. Sama ideja jest napraviti maketu koja bi mjerila odnos jačine potiska koji stvara određena elisa pri određenoj brzini u ovisnosti o vremenu u kojem se promatra odziv. Navedeno bi se izvelo pomoću mikroupravljača Atmega 328P (Arduino Nano) koji je reprogramibilan mikroupravljač što znači da se može više puta programirati odnosno reprogramirati. Rezultat bi bio prikazan pomoću Visual Studia pomoću kojega bi bilo napravljeno sučelje za upravljanje digitalnom vagom tj. motorićem.

1.1. Zadatak rada

Pomoću softverske i hardverske podrške napraviti digitalnu vagu na koju se može pričvrstiti trofazni motor s elisom te pomoću serijske komunikacije s računalom upravljati s istom. Sučelje treba sadržavati graf na kojem bi se iscrtavala ovisnost potiska i vremena potrebnog da se određena brzina odnosno potisak razvije, isto kao i potrebne tipke pomoću kojih se upravlja motorom.

2. MJERENJE STATIČKOG ELISNOG POTISKA

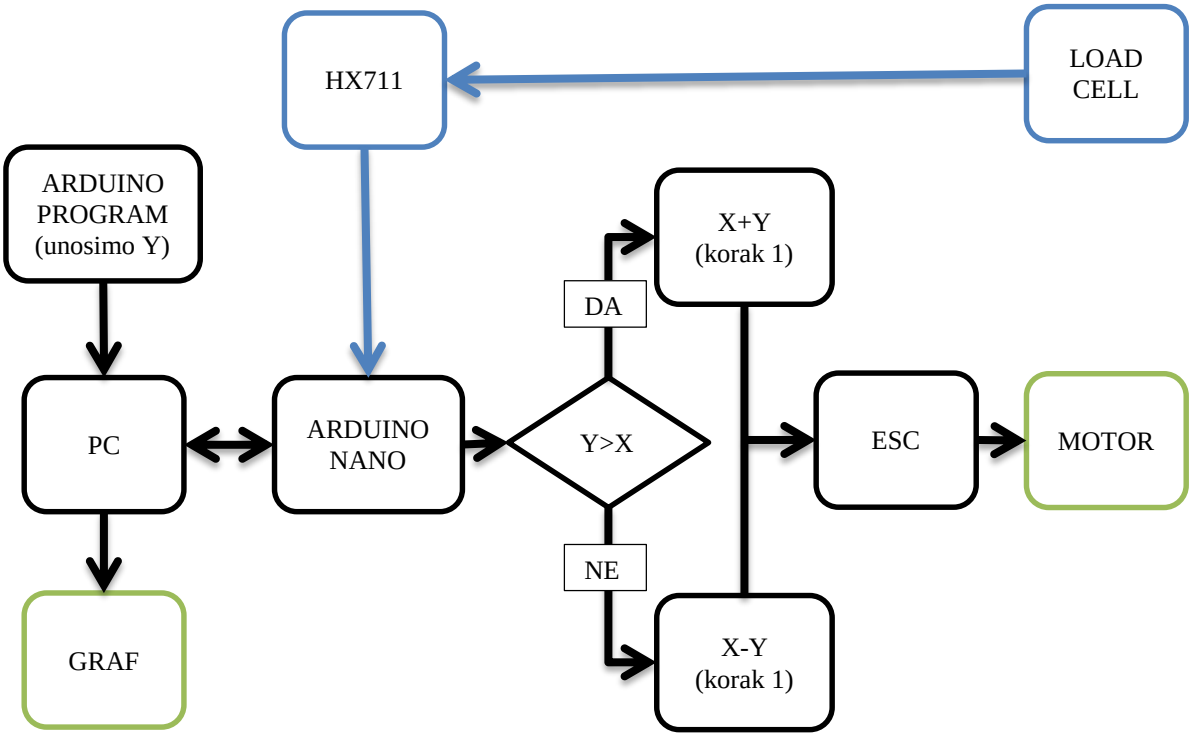
Elisni potisak predstavlja silu koju elisa uzrokuje svojom vrtnjom te je proporcionalna kutnoj brzini elise. Svojim zakretanjem djeluje na osovinu na koju je pričvršćena a samim time i na cijeli sustav koji se nadovezuje na osovinu odnosno motor. Elisni potisak ovisi o aerodinamici izrade same elise, broju krakova koji sadržava elisa, dijametru kojeg elisa opisuje prilikom svog zakretanja te brzini kojom se ista okreće. Najčešće se izrađuju elise s 2, 3 i 4 kraka jer se tako dobije najbolja učinkovitost (bespotrebno je praviti elisu s npr. 12 krakova, više materijala = veća masa) i vibracija koje se pojavljuju zbog neravnoteže npr. elise vjetroagregata, izrađuju se s tri kraka zbog vibracija prilikom naleta vjetra itd.. Samo mjerenje se izvodi iz tog razloga da bi se različite elise mogle uspoređivati jer izgled kao izgled elise nam ne može puno pomoći stoga se elise moraju testirati pri određenim uvjetima rada za koja su napravljena te usporediti i zaključiti koja od testiranih elisa daje najbolje performanse. Elisni potisak može se mjeriti i vertikalno i horizontalno, ovisno kako je uređaj za mjerenje potiska osmišljen (sl. 2.1.).



Sl. 2.1. Sustav za mjerenje elisnog potiska.

Kao što se vidi, elisa je zajedno s motorom pričvršćena na vrhu sustava. Motor je mehanički učvršćen kako bi silu koju proizvodi pomoću elise prenio na uređaj koji mjeri silu, a nalazi se u podnožju sustava. Također, sustav je mogao biti napravljen tako da potisak odnosno sila koju stvara elisa djeluje na sustav ali prema dolje tj. da je smjer sile zaokrenut za 180° u odnosu na silu koja je ucrtana na slici 2.1..

2.1. Upravljački sustav i algoritam



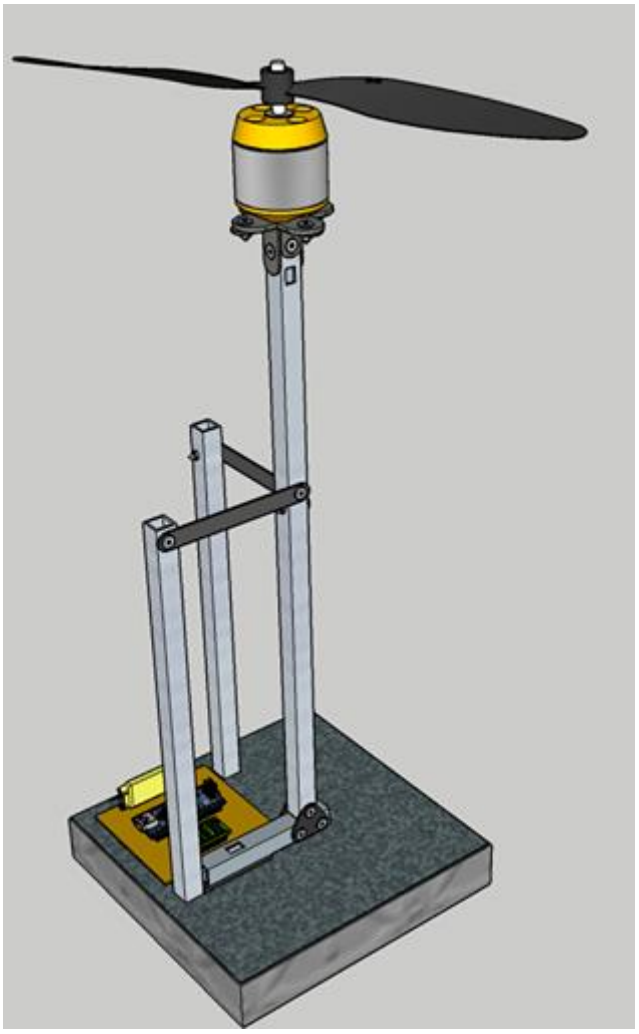
Sl 2.2. Funkcionalni blok dijagram

Na slici 2.2. vidi se pojednostavljeni algoritam rada digitalne vage za mjerenje statičkog elisnog potiska. Pomoću vanjskih jedinica (miš/tipkovnica) unosimo željene parametre (brzine, vremenski interval i dr.) u računalo koje preko sučelja napravljenog u Visual Studiu preko serijske komunikacije (USB-a) komunicira s Arduinom. Arduino uspoređuje primljene parametre s postavljenim parametrima te po potrebi primljene parametre stavlja u trenutne parametre

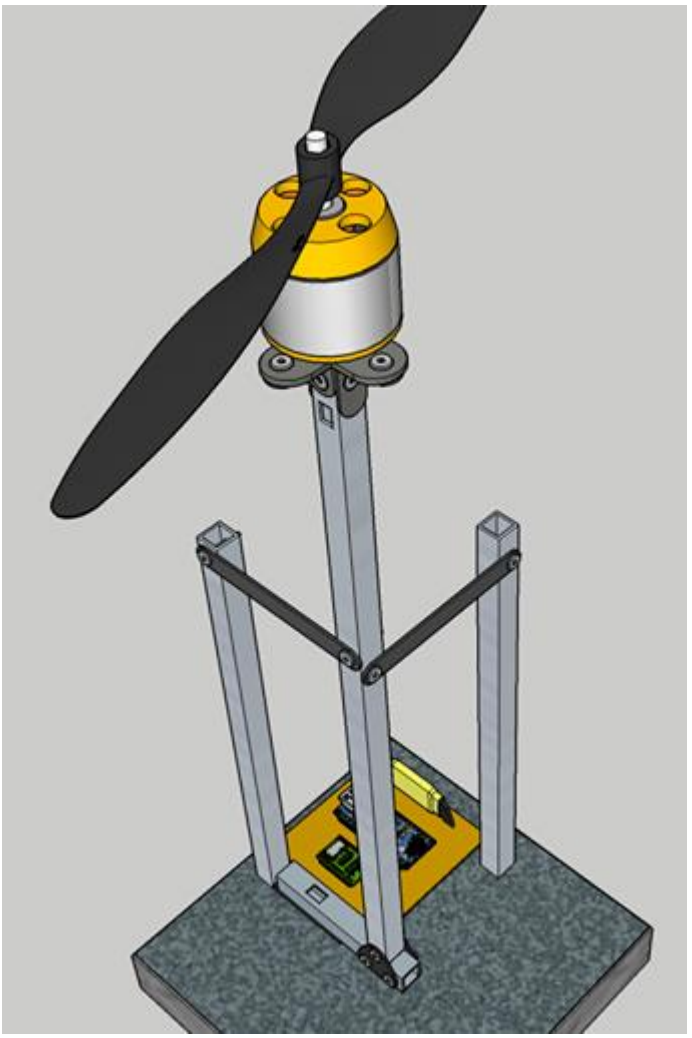
odnosno uspoređuje primljenu (postavljenu) brzinu koja je na dijagramu označena s Y te je uspoređuje s trenutnom brzinom X. Nakon uspoređivanja vrijednosti X i Y, poduzima potrebne radnje: ako je Y>X, na X dodaje se „1“ sve dok se vrijednosti X i Y ne izjednače, ako Y<X tada na X dodaje vrijednost „-1“ i to se ponavlja sve dok vrijednost X i Y nisu jednake. Isto tako, podatke primljene iz ćelije za mjerenje sile (engl. *Load cell*) šalje nazad računalu zajedno sa vremenom u kojem je sila izmjerena i trenutnom brzinom X. Računalo nakon svakog primanja paketa (u kom se nalazi vrijeme, potisak, brzina Y, i trenutna brzina X) iscrtava graf ovisnosti potiska o vremenu. Između računala i Arduina ostvarena je obostrana komunikacije odnosno Arduino kao i računalo i prima i šalje pojedine podatke i to na način o kojem će se kasnije detaljnije.

3. SUSTAV ZA MJERENJE STATIČKOG ELISNOG POTISKA

Prvo je potrebno osmisliti maketu odnosno izgled digitalne vage za mjerenje statičkog elisnog potiska što je učinjeno u 3D SketchUp programu (sl. 3.1., sl. 3.2.). Osmišljeno je tako da postolje stoji vertikalno te da elisa stvara potisak prema dolje odnosno nastoji uzdići vagu prema gore. Sve bi se nalazilo na drvenom postolju na kojem je pričvršćena vaga tj. load cell te se uz nju još nalaze i 3 metalna „stupa“ gdje bi jedan stup služio za povezivanje load cell-a i motora, a preostala dva stupa služe kao stabilizatori tog središnjeg stupa. Vaga ne smije biti male težine odnosno mase, jer bi je motor mogao uzdići te je predviđeno da je vaga oko 600 grama mase (za sve veće motore, vagu je potrebno dodatno učvrstiti). Na središnjem stupu trebao bi se nalaziti motor na koji je pričvršćena elisa, razlog zašto motor nije pričvršćen izravno na vagu jest taj što je tada elisa pre blizu postolju te je tako strujanje zraka ometano. Na drvenom postolju pričvršćeno je hardversko rješenje koje bi se sastojalo od Arduina (jezgre digitalne vage), HX711 A/D pretvornika, ćelije za mjerenje potiska te elektroničkog upravljača brzine koji služi za zakretanje (upravljanje) trofaznim motorom.



Sl. 3.1. 3D model makete



Sl. 3.2. 3D model makete (drugi kut)

3.1. Kotrišteni alati, hardwer i programska podrška

Za izradu makete koristio se sljedeći alat:

- Bušilica
- Lemilica
- Brusilica
- Kliješta
- Vijci
- Cin žica
- Vodikov peroksid i solna kiselina

Od hardwerskog dijela korišteni su sljedeći moduli:

- Računalo
- Arduino Nano
- Trofazni motor
- Pločica za jetkanje
- Elektronički upravljač brzine
- HX711 A/D pretvornik
- PC napajanje
- Ćelija za mjerenje sile

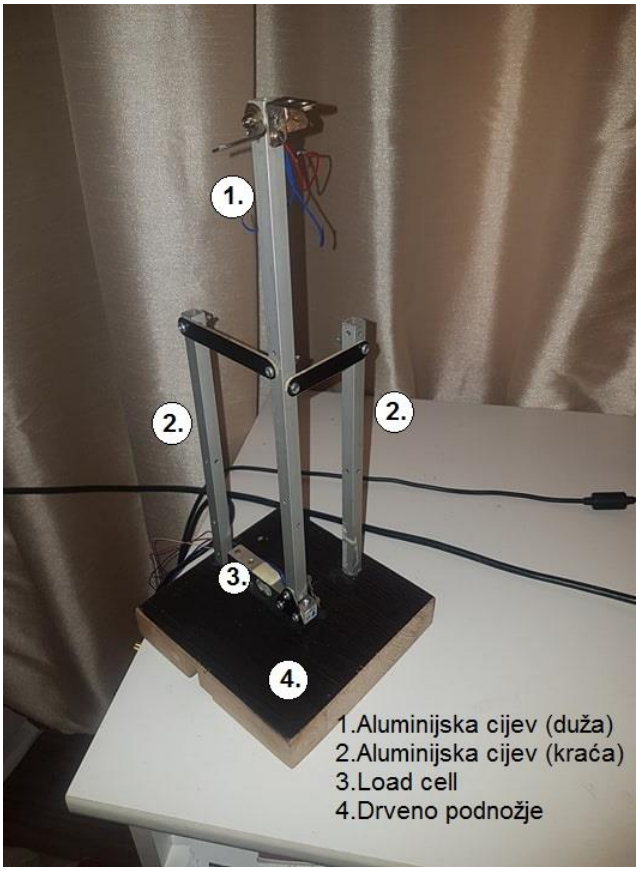
Potrebna programska podrška je sljedeća:

- Arduino IDE 1.8.4. (x64)
- Visual Studio 2012

Kod izrade makete bušilicom su napravljeni provrti na drvetu za metalne stupove i ćeliju za mjerenje sile, te je svrdlom od 1 mm bušena pločica na koju su kasnije lemilicom i cin žicom polemljeni elementi. Kliještima su pričvršćeni potrebni vijci na maketi i skidana je izolacija. Vodikovim peroksidom i solnom kiselinom napravljena je mješavina pomoću koje je uklonjena nepotrebna površina bakra na otiskanoj pločici (detalji o jetkanju kasnije). Računalo je potrebno radi izrade softverskog djela makete u programima Arduino IDE 1.8.4. (x64) i Visual Studio 2012 te se dio programa isprogramirao na Arduino Nano pločicu dok je ostatak ostao na računalu za izgled upravljačkog sučelja. Trofazni motor upravljan je elektroničkim upravljačem brzine, ćelija za mjerenje sile je preko HX711 A/D pretvornika spojena na Arduino te se svi hardverski elementi napajaju preko PC napajanja.

3.2. Mehanički ustroj (fizički izgled i konstrukcija)

Za postolje makete korišteno je drvo dimenzija 150 x 150 x 25 mm. Na drvo je pričvršćena aluminijska konstrukcija koja sadrži dvije fiksne cijevi četvrtastog presjeka dimenzija 200 x 10 x 10 mm te jedna koja je učvršćena na vagu dimenzija 300 x 10 x 10 mm (vidi sliku 3.3.). Na dulju aluminijsku cijev koja je pričvršćena na vagu, postavlja se trofazni motorić na čiju osovinu je pričvršćena elisa koja bi se trebala ispitivati. Središnja cijev stabilizirana je s dvije fiksne cijevi koje se nalaze kraj nje te služe da središnja cijev ima samo translaciju po vertikalnoj osi. Na drvenoj podlozi postavljena je i pločica na kojoj su polemljeni potrebni hardverski elementi (sl 3.4.). Maksimalno opterećenje vage iznosi 10 kg što je dovoljno za ispitivanje motora koji se koriste u npr. konvencionalnim dronovima.



Sl. 3.3. Izgled kontrukcije digitalne vage

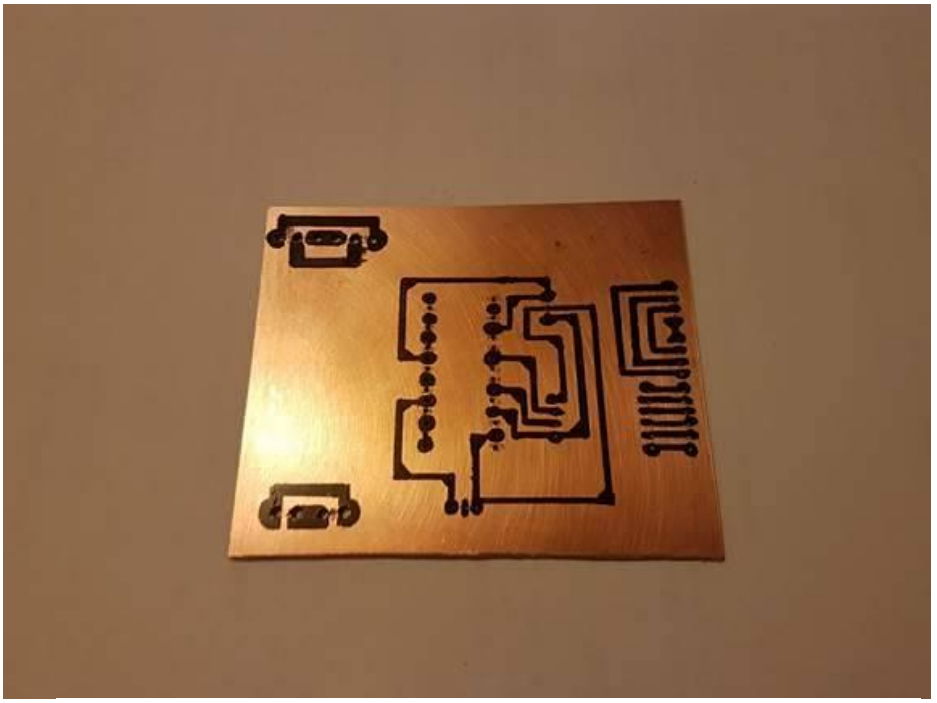
Sl. 3.5. Završni izgled digitalne vage za mjerenje elisnog potiska

3.3. Izrada tiskane pločice

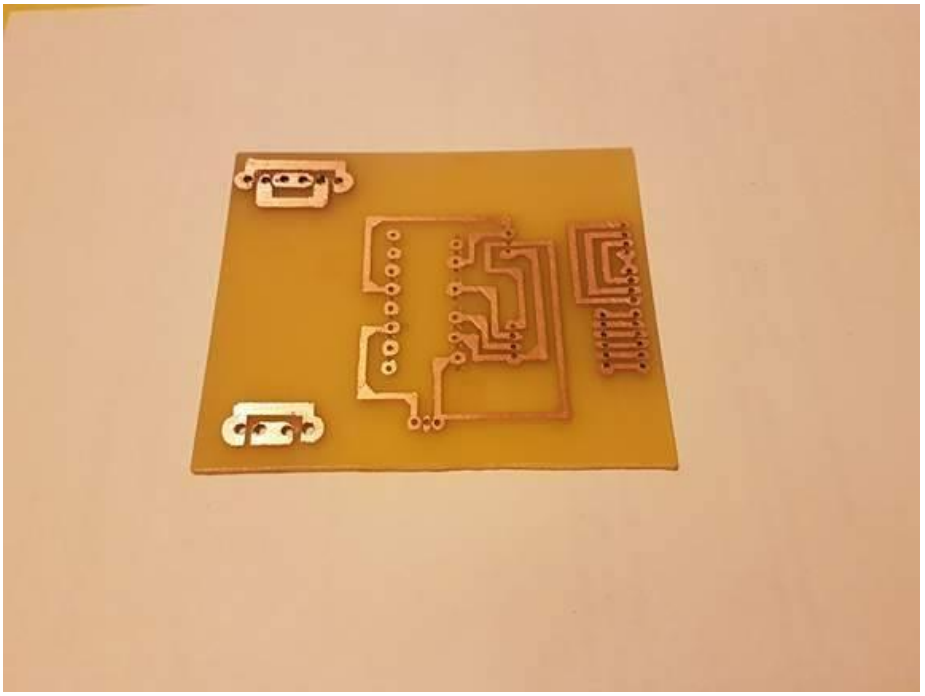
Tiskana pločica izrađena je u računalnom programu EAGLE, no prije toga potrebno je osmisliti međusobnu povezanost korištenih modula. EAGLE pruža mogućnost odabira potrebnih elemenata te automatskog raspoređivanja istih na tiskanu pločicu radi smanjenja prostora i izbjegavanja dodirivanja vodova. Nakon virtualne izrade tiskane pločice, potrebno je prenijeti istu na bakrenu tiskanu pločicu pomoću nekog od postupka (osvjetljavanje, ručno crtanje, prenošenje peglom i dr.)(sl. 3.6.).

Proces jetkanja se vrši kada je shema iz EAGLE-a prenesena na bakrenu pločicu, i to tako da se pločica za jetkanje ubaci u tekućinu H_2O_2 koja se sastoji od solne kiseline, vodikovog peroksida (hidrogena) i vode i to u omjeru 0.2 : 0.15 : 0.5 na vremenski period od cca. 15-ak minuta (ovisno o koncentraciji hidrogena). Jetkanjem se uklanja nezaštićeni dio bakrene površine s pertinaksa te ostavljeni dio predstavlja vodove koji služe za povezivanje modula (sl. 3.7.).

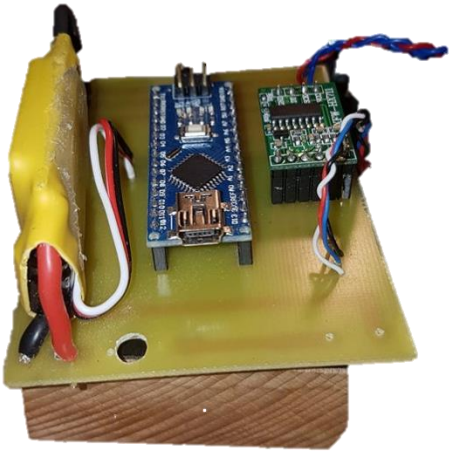
Izjetkanu pločicu tada je potrebno izbušiti sa svrdlom promjera 1 mm te se tada postavljaju potrebni moduli i to pomoću lemljenja (sl. 3.8.). Lemljenje je proces pomoću kojega se komponente poput otpornika, tranzistora, dioda itd. „pričvršćuju“ za bakrenu površinu odnosno vodove i to pomoću cina koji ima vrlo nisko talište te je vrlo dobar vodič. Prilikom lemljenja, potrebno je očistiti željena mjesta spajanja i pripaziti da se komponente koje se leme ne pregrijavaju previše jer može doći do njihova uništenja.



Sl. 3.6. Otiskana pločica



Sl. 3.7. Izjetkana i izbušena pločica



Sl. 3.8. Polemljena pločica

3.4. Komponente i njihov opis

U tablici 3.1 je popis komponenata korištenih za izradu makete digitalne vage za mjerenje dinamičkog elisnog potiska .

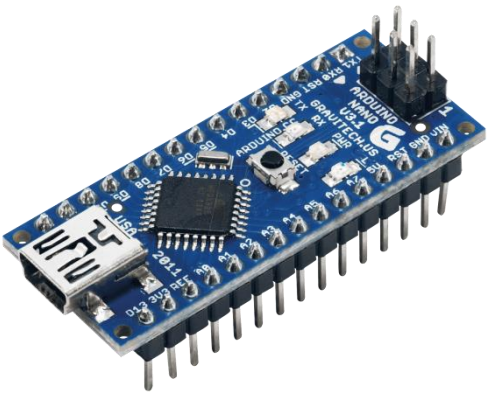
Naziv	Količina	Cijena
Arduino nano ATmega 328P	1	15 kn
10 kg Load cell	1	20 kn
Trofazni motor	1	70 kn
HX711 A/D pretvarač	1	6,50 kn
ESC-30A	1	40 kn
Drvena podloga	1	5 kn
Pločica 160 x 100 mm	1	16 kn

Hidrogen	150 (mL)	9 kn
Solna kiselina	1 (L)	13 kn
Aluminijske cijevi	1 (m)	15 kn
		Ukupno: 209,50 kn

Tablica 3.1. Popis komponenata i njihova cijena

3.5. Arduino Nano

Arduino Nano je pločica koja sadrži softversku i hardversku platformu koja je vrlo jednostavna za programiranje i ima jako široko područje primjene. Sastoji se od mikrokontrolera ATMEGA 328P, mini USB priključka, digitalnih i analognih pinova kao i pinova napajanja i uzemljenja (njih sveukupno 30), tipke za reset mikrokontrolera i ICSP porta. Svi analogni i digitalni pinovi mogu se koristiti i za slanje i za primanje podataka iz/u Arduino. Detaljnije specifikacije nalaze se u prilogu 1..

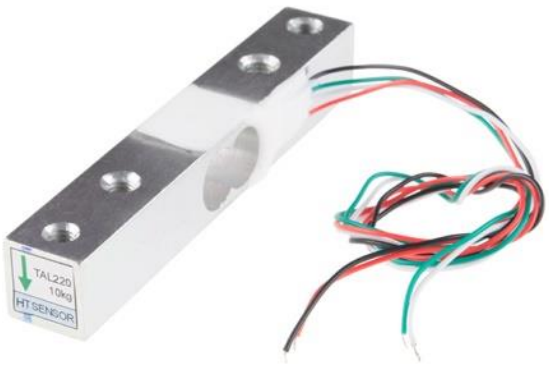


Sl. 3.9. Arduino Nano

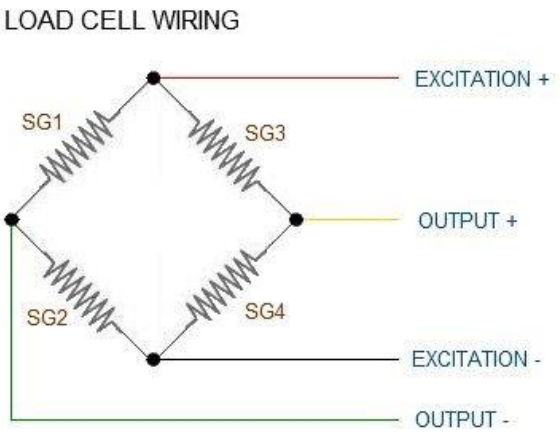
3.6. Ćelija za mjerenje sile

Ćelija za mjerenje sile (engl. *Load Cell*) predstavlja uređaj koji mehaničku energiju pretvara u električni signal (Sl. .). Imamo više vrsta ćelija tj. raznih principa djelovanja npr. pomoću Wheatstoneova mosta ili na principu Piezo-električkog efekta. Princip Piezo-električnog efekta zasniva se na stvaranju naboja prilikom djelovanja sile na kristalnu rešetku te se pomoću tog naboja određuje jakost sile koja djeluje na ćeliju. Naboj je naravno proporcionalan sili koja djeluje, te se ćelija za mjerenje prvenstveno mora kalibrirati kako bi za jedinicu sile proizveo određenu promjenu naboja.

Ćelija za mjerenje također može biti izvedena i pomoću Wheatstoneova mosta (sl. 3.11.) i to tako da nepoznat otpornik predstavlja ćelija a preostala tri su nam poznata. Djelovanjem sile na ćeliju, njen otpor se mijenja te se Wheatstoneov most dovodi u neravnotežu odnosno imamo razliku potencijala i prema toj razlici potencijala možemo odrediti koliki je iznos sile koja djeluje na ćeliju. Ćelija za mjerenje sile se preko analogno-digitalnog pretvornika HX711 spaja na Arduino.



Sl. 3.10. Ćelija za mjerenje sile



Sl. 3.11. Wheatston-ov most

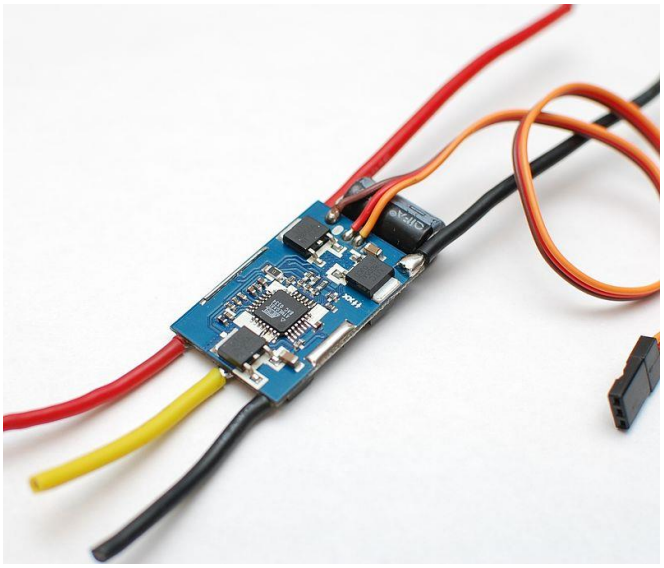
3.7. HX711 A/D pretvarač

HX711 jest 24-bitni analogno-digitalni pretvarač kako mu i sam naziv kaže, koristi se za pretvaranje analognog signala koji se dobije iz load cell-a u digitalni signal prilagođen za ulaz u Arduino Nano (sl. 3.12.). Sadrži 10 nožica od kojih se 4 nožice koriste za napajanje i komunikaciju s Arduinom a ostalih 6 služe za spajanje load cell-a.

Sl. 3.12. HX711 A/D pretvarač

3.8. Elektronički upravljač brzine – 30 A

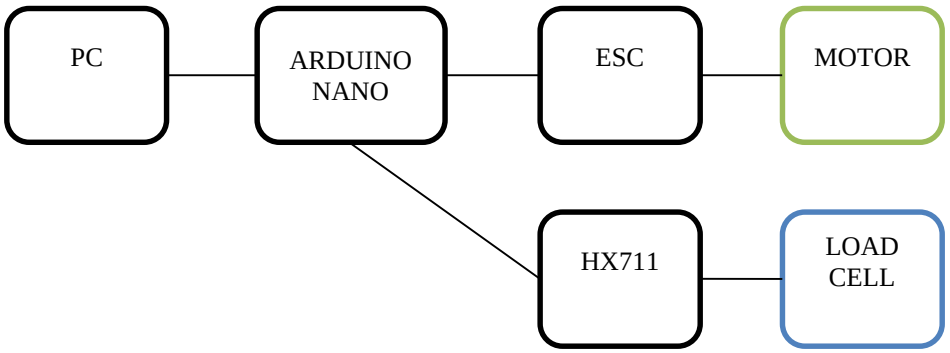
Elektronički upravljač brzine (ESC - engl. *electronic speed controler*) motora bez klizećih prstenova i četkica, oznaka 30 A označava da se radi o ESC-u maksimalne struje od 30 A (ampera). U ovom radu koristi se motor od 1000 kV što znači da se njegova brzina vrtnje poveća za 1000 rpm za svaki dodan volt. ESC-ova najčešća upotreba jest kod RC (engl. *remote control*) uređaja kao i kod dronova jer se pomoću istosmjerne struje iz baterije treba upravljati motorićima koji mogu mijenjati svoju brzinu jer su trofazni pa se manipuliranjem magnetskog polja dobije željena brzina. ESC sadrži 8 žica od kojih su 3 ulazne, a preostalih 5 su izlazi (Sl. 3.13.). Dvije su napajanje ESC-a, tri su za pojedinu fazu trofaznog motora, jedna je za kontrolu ESC-a, jedna je +Vcc (5V) dok je druga uzemljenje i služe za napajanje npr. vanjskog WiFi odašiljača pomoću kojega se daljinski upravlja ESC-om.



Sl. 3.13. Elektronički upravljač brzine (ESC)

3.9. Električna i blokovska shema sustava

Izgled blokovske sheme digitalne vage (sl. 3.14.).

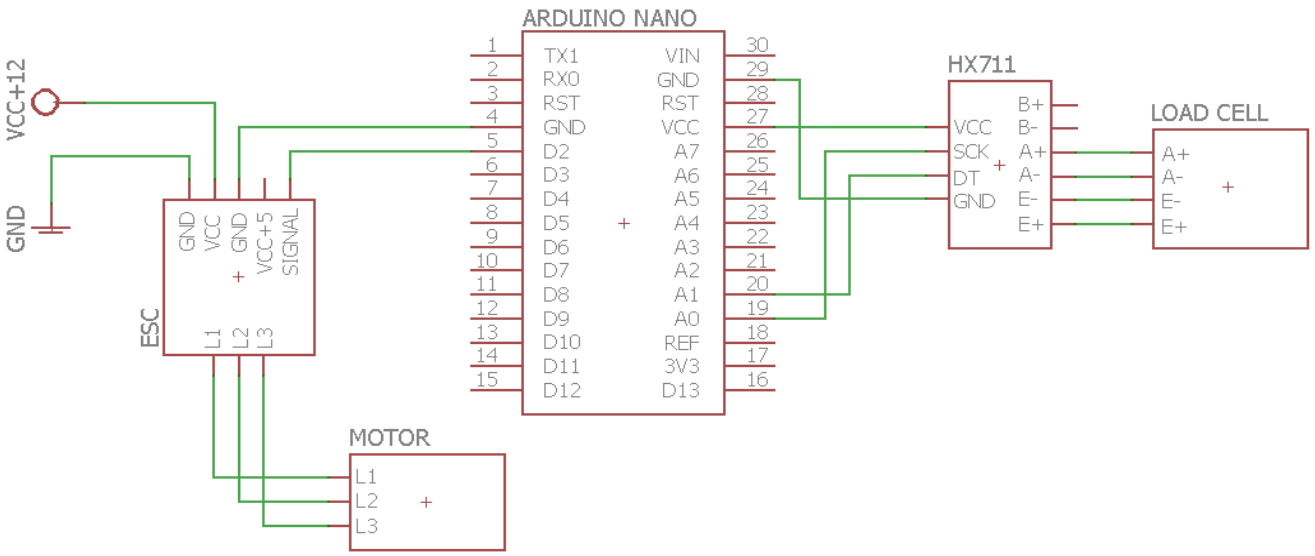


Sl. 3.14. Blok dijagram električne sheme

gdje svaki od blokova predstavlja pojedini hardver koji je korišten. Linija koja povezuje blokove međusobno predstavlja stvarnu povezanost hardverskog djela žicom (jednom ili više). Blok „PC“ predstavlja računalo preko kojega se uz pomoć izrađenog sučelja na njemu i vanjskih ulazno-izlaznih jedinica upravlja Arduinom. Blok „Arduino Nano“ predstavlja „mozak“ digitalne vage za mjerenje elisnog potiska te se na njemu nalazi mikrokontroler koji je isprogramiran tako da se posredstvom Arduina upravlja motorom i prikupljaju se podaci. „HX 711“ jest analogno-digitalni

pretvornik pomoću kojeg Arduino komunicira s digitalnom vagom. Ćelija za mjerenje sile jest vaga koja daje vrijednosti sile kojom se djeluje na nju. Blok „ESC“ predstavlja elektronički kontroler brzine koji je povezan s Arduinom i trofaznim motorom te istosmjerni impuls dobiven iz Arduina pretvara u trofazni izmjenični signal pomoću kojega se upravlja motorom. „Motor“ jest blok koji predstavlja trofazni motor na kojem se nalazi elisa kojoj se mjere karakteristike.

Detaljna električna shema (sl. 3.15.).



Sl. 3.15. Električna shema Digitalne vage za mjerenje statičkog elisnog potiska

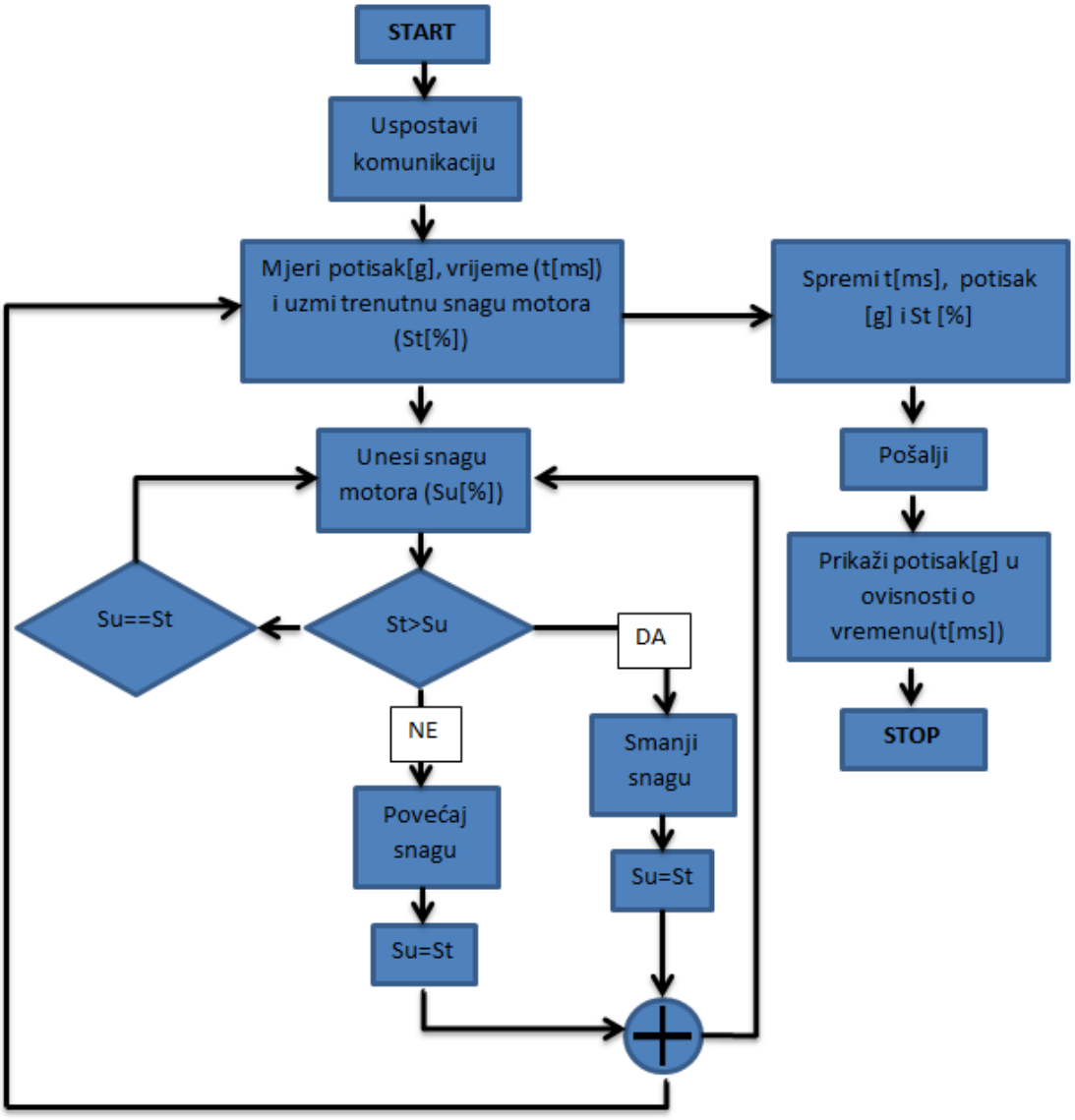
Elektronički upravljač brzine jest spojen na napajanje od PC računala, na trofazni motor i na Arduino kako je prikazano na shemi. Na Arduino Nano spojeni su još i HX711 A/D pretvornik s jednim upravljačkim, dva pina za napajanje i jednim pinom za primanje informacija. Ćelija za mjerenje sile je spojena samo na HX711 A/D pretvornik koji joj je ujedno i napajanje te se preko A+, A-, E+ i E- pinova, dobiva stvoreni potisak.

3.10. Algoritam upravljanja

Nakon spajanja potrebnih hardverskih elemenata, potrebno je izraditi potrebno softversko sučelje i program kako bi se moglo upravljati digitalnom vagom. Softverska podrška izrađena je u C programskom jeziku i to u programu Arduino IDE programu koje je originalno i vrlo jednostavno za programiranje.

Prvo je potrebno definirati biblioteke koje su nam potrebne u daljnjem programiranju te nam olakšavaju i ubrzavaju programiranje određenog djela koda pošto je u njima unaprijed određen tj. isprogramiran rad pojedinih elemenata, korištene biblioteke nalaze se u prilogu 2.1 i 2.2. U bibliotekama je već unaprijed spremljen rad HX 711 modula isto kao i rad ESC-a, a nalaze se pod nazivom <Servo.h> i <hx711.h>. Najbitnije je definirati ulazno/izlazne (IO) pinove odnosno pinove koji će služiti za primanje a koji za slanje podataka, bez toga Arduino neće dobro funkcionirati.

Na slici 3.16. nalazi se blog dijagram algoritma koji opisuje tok radnji i podataka. Početak jest zapravo uspostava komunikacije između računala i Arduina preko USB serijskog porta. Nakon uspostave komunikacije postavlja se vrijednost brzine na koju motor treba biti postavljen i to u postotcima (0%-motor je zaustavljen, 50%-motor je na pola svoje brzine/snage, 100%-motor radi punom snagom/brzinom). Ako je unesena snaga „S_u“ manja od trenutne snage motora „S_t“ tada se grananje svodi na „DA“ te se snaga smanjiva sve dok se trenutna snaga S_t ne izjednači s unesenom snagom S_u. Isto tako, moguće je da je unesena snaga S_u veća od trenutne snage motora što vodi ka grananju „NE“ te se trenutna snaga S_t povećava sve dok ne dostigne vrijednost unesene snage S_u. Ako je unesena snaga S_u jednaka trenutnoj snazi S_t tada se ništa ne događa sve dok se ne unese nova snaga S_u. Svaki korak izmjene trenutne vrijednosti snage S_t rezultira s prikupljanjem podataka s ćelije za mjerenje sile tj. svakim povećanjem/smanjenjem snage, formira se string od 4 podatka u kojem je sadržano vrijeme, potisak, unesena brzina, trenutna brzina o čemu će se kasnije nešto detaljnije.



Sl. 3.16. Blok dijagram algoritma digitalne vage za mjerenje elisnog potiska

3.11. Komunikacija i HM sučelje

Komunikacija je ostvareno preko serijskog porta odnosno USB porta. Podaci koji dolaze u s Arduina poslani su kao string odnosno polje koje sadži 4 pod polja od koje svako to polje nosi po jedan podatak (sl. 3.17.). Prvo mjesto u polju zauzima vrijeme početka komunikacije, drugo jest zauzeto potiskom koje je primljeno sa ćelije za mjerenje sile, treće predstavlja postavljenu brzinu dok posljednje polje predstavlja trenutno postavljenu brzinu (sl. 3.18). Brzina je predstavljena u postotcima iz tog razloga što nemaju svi motori isto upravljanje, u ovom slučaju 0% šalje električkom upravljaču brzine podatak „29“ dok 100% iznosi „150“ te on upravlja signalom koji šalje pojedinoj fazi motora. Potisak je izražen u gramima a vrijeme je izraženu u milisekundama.

```
String zajedno = String (vrijeme) + "/" + String(potisak) +
"/" + String(throttle) + "/" + String(currentThrottle);
Serial.println(zajedno);
```

Sl. 3. 17. Izgled stringa koji se šalje sa Arduina

Slika 3.18. Izgled paketa podataka koji se šalji serijskim portom

Brzina komunikacije Arduina s računalom preko USB serijskog porta iznosi 9600 baud-a odnosno 9600 bita u sekundi što je definirano u Arduinu, naravno postoje i mnogo veće/manje brzine od navedene no u ovom radu 9600 je dovoljno.

Podatke koji su poslani s Arduina primati ćemo pomoću sučelja koje će biti napravljeno u Visual Studiu 2012. Potrebno je incijalizirati port u Visual-u a to se čini na sljedeći način:

```
private void InitializePort(){    // inicijalizacija porta

      myport = new SerialPort();   // definiranje serijskog porta
      myport.BaudRate = 9600;  // definiranje brzine prijenosa podataka
      myport.Parity = Parity.None;    // definiranje kontrolnih bitova, netreba,
      myport.DataBits = 8; // veličina dolaznog podatka definira se na 8 bitova
      myport.StopBits = StopBits.One; // definiranje broja bitova za prekid (1)

myPort.DataReceived += new SerialDataReceivedEventHandler(ReceivedSerialHandler);
      // definiranje dohvačenih podataka
      myport.Open(); // otvaranje porta
}
```

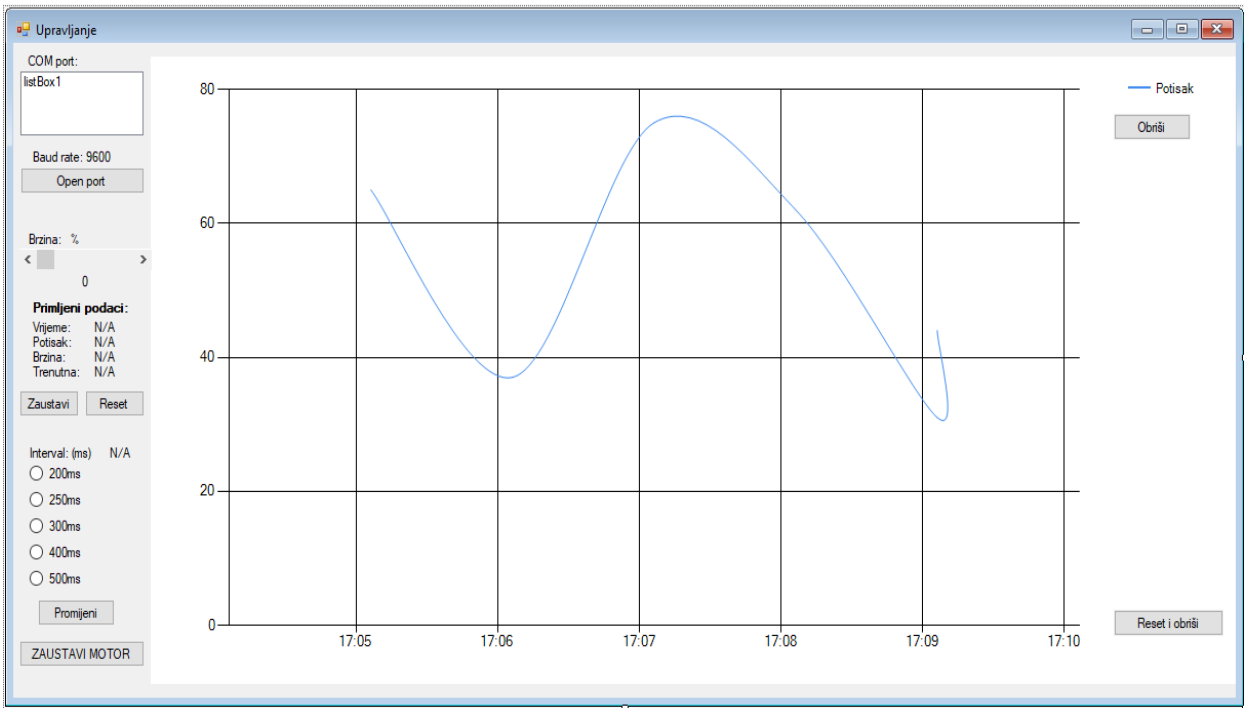
Ili na ovaj način:

```
myPort = new SerialPort(comPort, baudRate, Parity.None, 8, StopBits.One);
```

Obje inicijalizacije daju isti rezultat.

3.12. HM sučelje

Sučelje je izrađeno u programu Visual Studio 2012 koji na početku prikaže samo prazan prozor kojemu se prvo određuje veličina, nakon toga dodajemo potrebnu dugmad, tekstove, radio buttone, prozorčiće za unos podataka, dijagrame itd. Izradba samog sučelja vrlo je jednostavno, no problem predstavlja daljnje povezivanje sučelja s nekim drugim sučeljem (u ovom slučaju Arduino INO sučelje).



Sl. 3.19. Izgled izrađenog sučelja pomoću Visual Studi-a

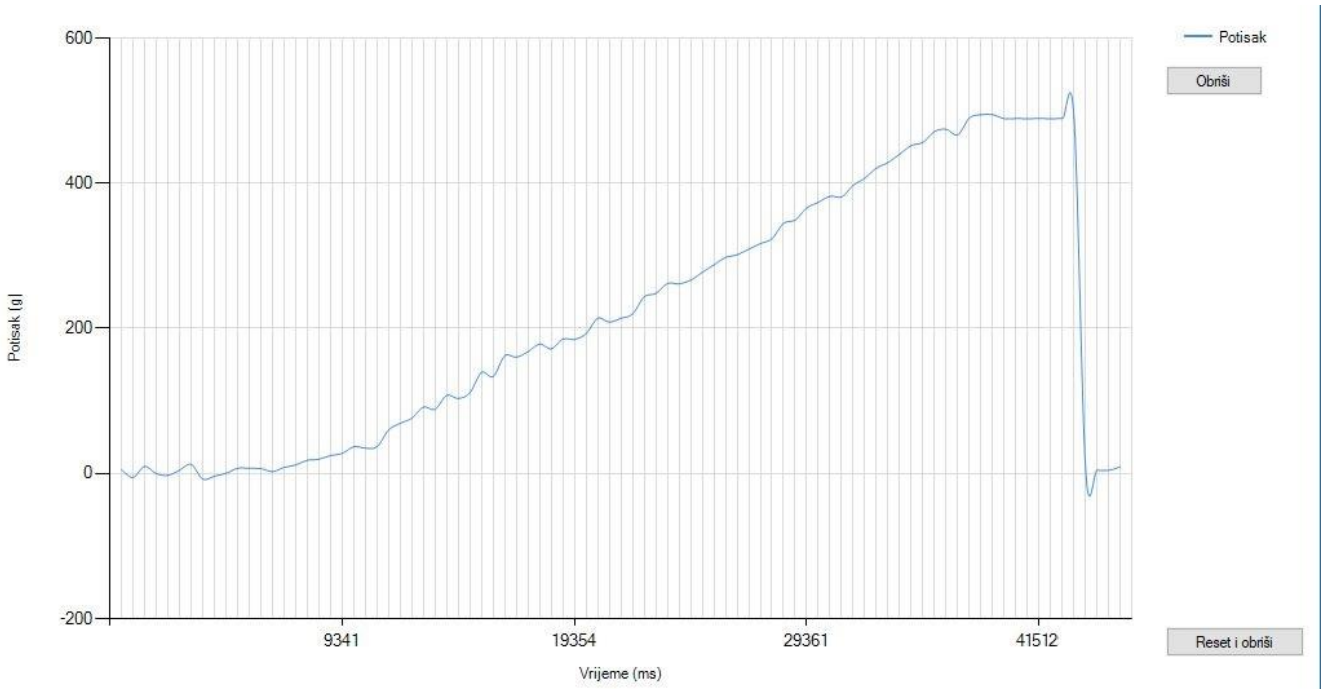
Izgled sučelja može se vidjeti na slici 3.19. Prozor „COM port:“ predstavlja izlistanje portova koji se koriste na računalu, te se odabirom na port i pritiskom na gumb „Open port“ ostvaruje povezivanje komunikacije preko tog porta s uređajom spojen na isti te se gumbu mijenja naziv u „Port opened“ te on više nije u funkciji kako se nebi moglo dva puta kliknuti na ostvarivanje komunikacije. Gumb „Obriši“ obriše graf, te se nastavlja ponovno iscrtavati dok gumb „Reset i obriši“ briše graf i vraća brzinu motorića na 0% s korakom od -1. Gumb „Zaustavi“ zaustavlja iscrtani graf dok gumb „Reset“ resetira vrijeme. Radio button-i kod „Interval (ms)“ predstavljaju vrijeme između svake iscrtane točke na grafu. Gumb „ZAUSTAVI MOTOR“ šalje motoriću vrijednost 0% tj. 29 te se motor zaustavlja skoro pa trenutno (zbog inercije elise).

4. TESTIRANJE I REZULTATI

Digitalnom vagom mogu se testirati statičke i dinamičke karakteristike pojedine elise no ovim putem obradit će se statička mjerenja elise. Statička mjerenja izvode se tako da se na motor stavi elisa koju želimo testirati, nakon toga se preko izrađenog sučelja postepeno povećava brzina motora sve dok se ne postigne maksimalna brzina te se dobije odziv odnosno graf ovisnosti potiska koji stvara postavljena elisa i vremena u kojem se događala promjena brzine. Graf se može spremiti te se testiranje može izvoditi ponovo no ovaj puta s nekom drugom elisom te se prijašnji graf usporedi s novim grafom. Usporedba služi kako bi se moglo odrediti koja elisi ima bolje performanse u određenim uvjetima za koje su predviđene.

4.1. Testiranje elise s 2 kraka

U nastavku možemo vidjeti graf dobiven pomoću izrađene digitalne vage i to s elisom koja ima 2 kraka, promjera 122 mm (slika 4.1.)



Sl. 4.1. Statičke karakteristike elise s 2 kraka, radijusa 122 mm

Na slici se može vidjeti da ukupno trajanje iznosi nešto više od 40 sekundi no motor je krenuo ubrzavati u 8. sekundi te je maksimalnu brzinu postigao u 37. sekundi nakon ostvarivanja komunikacije. Maksimalni potisak koji je elisa napravila iznosi oko 480 grama i to pri maksimalnoj brzini 37. sekundi. Vide se male oscilacije no to je isključivo zbog velike osjetljivosti load cell-a.

6. ZAKLJUČAK

Cilj završnog rada bio je izraditi maketu digitalne vage za mjerenje elisnog potiska. Za izradu vage, samo kodiranje sustava te mjerenja, bilo je potrebno poznavati teorijsku podlogu vezanu za kodiranje Arduina i način funkcioniranja korištenih sklopova. Glavni odnosno upravljački dio ovog rada jest Arduino Nano pomoću kojega je povezano sve sklopovlje te se preko njega komunicira s računalom i sklopovljem. Elektronički upravljač brzine je integrirani sklop koji služi za upravljanje trofaznim elektromotorom i to na taj način da pomoću istosmjerne struje stvara izmjenično magnetsko polje potrebno za ubrzavanje odnosno usporavanje elektromotora.

Usporedimo li dobivena mjerenja s analizom rada, primjećuje se sličnost dobivenih rezultata što znači da je odstupanje neznatno i neizbježno zbog pojednostavljivanja i zanemarivanja raznih parametara. Primjećuje se da je elisni potisak ovisan o brzini vrtnje samog elektromotora te o obliku, veličini i broju krakova elise.

Digitalna vaga za mjerenja elisnog potiska nema široku primjenu, no praktična je za testiranje i usporedbu raznih elisa te odabir pogodne za korištenje u određenim uvjetima.

LITERATURA

[1] Arduino IDE program
https://www.arduino.cc/en/main/software, 15.04.2017.

[2] Arduino HX711 A/D pretvornik
http://arduinotronics.blogspot.hr/2015/06/arduino-hx711-digital-scale.html, 15.04.2017.

[3] Arduino ESC
https://gist.github.com/col/8170414, 23.05.2017.

[4] Dino Car, diplomski
https://bib.irb.hr/datoteka/776579.diplomski_Zavrna_verzija_DinoCarFESB.pdf, 16.06.2017.

[5] HX711 datasheet
https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf, 23.06.2017

SAŽETAK

Kako bi se riješio problem usporedbe različitih elisa, izrađena je maketa digitalne vage za mjerenje statičkog elisnog potiska. Hardverski dio se sastoji od Arduina, ćelije za mjerenje sile (engl. Load cell), elektroničkog upravljača brzine (engl. ESC), analogno-digitalnog pretvornika HX711 te trofaznog motora. Softverski dio izrađen je u Arduino IDE i Visual Studio 2012 programima. Prije izrade same makete, bilo je potrebno proučiti koje sklopovlje je najpogodnije za izradu iste. Testiranja je moguće vršiti s više različitih elisa i trofaznih motora te se tako međusobnom usporedbom odabiru najpogodnije elise i motori u predviđenim uvjetima za rad.

Ključne riječi: ESC, digitalna vaga, elisni potisak, HX711

ABSTRACT

Digital scale for static airscrew boost measurements

To solve the comparison problem between different airscrews, a digital scale model for static airscrew boost was designed. The hardware consists of an Arduino board, a load cell, an ESC, the HX711 ADC and a three-phase engine. The software was designed in Arduino IDE and Visual Studio 2012. Before constructing the actual model, it was necessary to determine what hardware was optimal for this task. The comparison can be done with multiple airscrews and three-phase engines and thus it's possible to pick optimal combinations for the given work conditions based on these comparisons.

Keywords: ESC, Load cell, airscrew boost, HX711

ŽIVOTOPIS

Eugen Šimara rođen je 20.02.1996. godine u Osijeku. U Viljevu završava "Osnovna škola Ante starčevića – Viljevo", nakon čega upisuje srednju školu zanimanja elektrotehničara u Slatini "Srednja škola Marka Marulića Slatina". Tokom srednjoškolskog obrazovanja sudjeluje u županijskom natjecanju iz fizike. Postiže 1. mjesto na županijskom natjecanju u Virovitici te 1. mjesto na regionalnom natjecanju u Varaždinu u atletici, disciplina – bacanje kugle. Po završetku srednjoškolskog obrazovanja, upisuje preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Na drugoj godini se opredjeljuje za smjer Komunikacije i informatika.

U Osijeku, lipanj 2017. Eugen Šimara

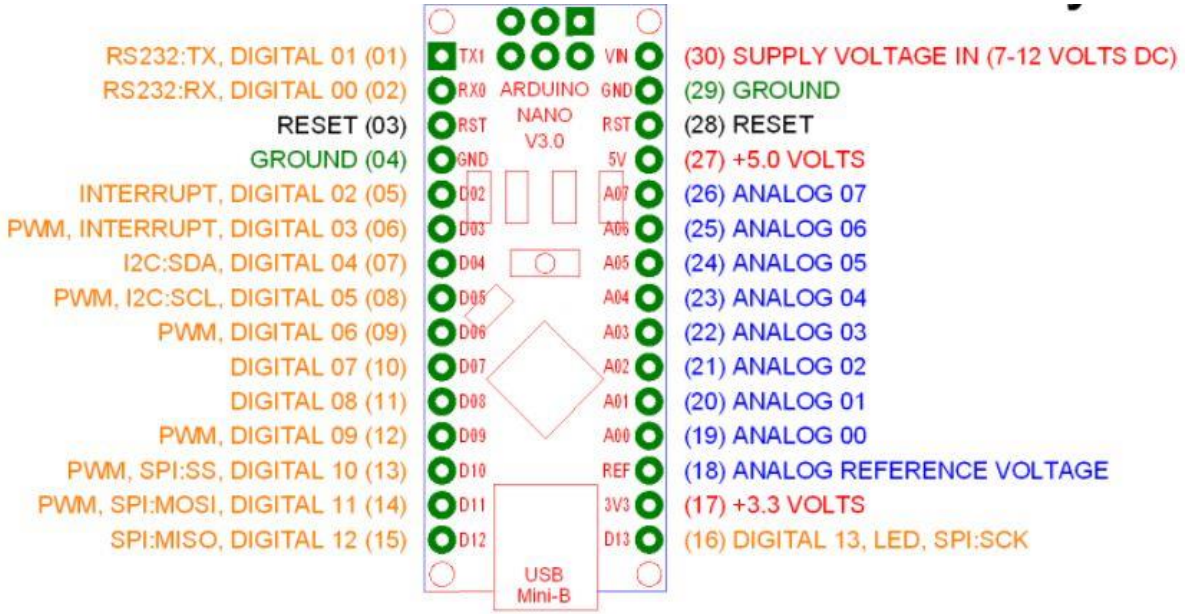
_____

(Vlastoručni potpis)

PRILOZI

Prilog 1: ATMega328p specifikacije

Mikrokontroler	ATmega328
Operativni napon	5V
Ulazni napon (preporučeno)	7 – 12 V
Ulazni napon (ograničenja)	6 – 20 V
Digitalni ulazno/izlazni pinovi	14
PWM kanali	6
Analogni ulazni pinovi	8
DC struja po ulaz/izlaz pinu	40 mA
Flash memorija	32 KB (ATmega 328) od čega 2 KB koristi za pokretanje sustava (bootloader)
SRAM	2 KB
EEPROM	1 KB
Brzina sata	16 MHz
Veličina	1.85cm x 4.3cm



Prilog 2.1: Hx711 library

```
Hx711::Hx711(uint8_t pin_dout, uint8_t pin_slk) :
		_pin_dout(pin_dout), _pin_slk(pin_slk)
{
	pinMode(_pin_slk, OUTPUT);
	pinMode(_pin_dout, INPUT);

	digitalWrite(_pin_slk, HIGH);
	delayMicroseconds(100);
	digitalWrite(_pin_slk, LOW);

	averageValue();
	this->setOffset(averageValue());
	this->setScale();
}

Hx711::~Hx711()
{

}

long Hx711::averageValue(byte times)
{
	long sum = 0;
	for (byte i = 0; i < times; i++)
	{
		sum += getValue();
	}

	return sum / times;
}

long Hx711::getValue()
{
	byte data[3];

	while (digitalRead(_pin_dout))
		;

	for (byte j = 3; j--;)
	{
		for (char i = 8; i--;)
		{
			digitalWrite(_pin_slk, HIGH);
```

```
                        bitWrite(data[j], i, digitalRead(_pin_dout));
                        digitalWrite(_pin_slk, LOW);
                }
        }

        digitalWrite(_pin_slk, HIGH);
        digitalWrite(_pin_slk, LOW);

        data[2] ^= 0x80;

        return ((uint32_t) data[2] << 16) | ((uint32_t) data[1] << 8)
                        | (uint32_t) data[0];
}

void Hx711::setOffset(long offset)
{
        _offset = offset;
}

void Hx711::setScale(float scale)
{
        _scale = scale;
}

float Hx711::getGram()
{
        long val = (averageValue() - _offset);
        return (float) val / _scale;
}
```

Prilog 2.2: Hx711 library

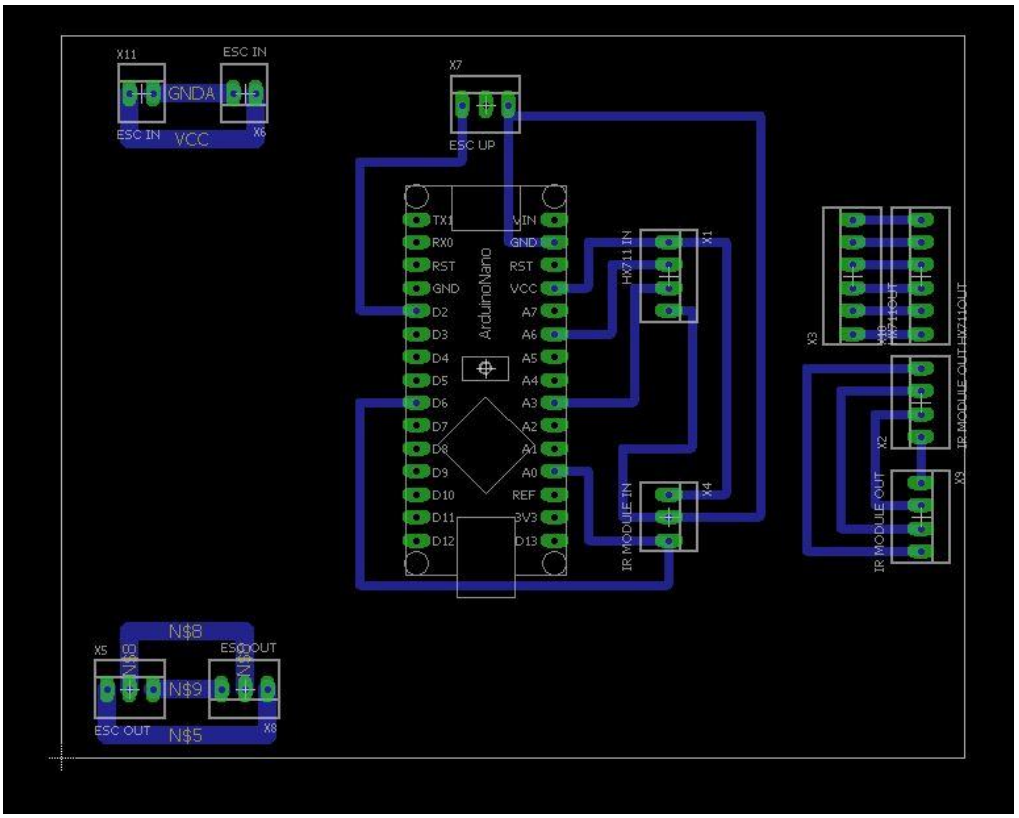
```
#ifndef HX711_H_
#define HX711_H_

#include "Arduino.h"

class Hx711
{
public:
	Hx711(uint8_t pin_din, uint8_t pin_slk);
	virtual ~Hx711();
	long getValue();
	long averageValue(byte times = 32);
	void setOffset(long offset);
	void setScale(float scale = 742.f);
	float getGram();

private:
	const uint8_t _pin_dout;
	const uint8_t _pin_slk;
	long _offset;
	float _scale;
};

#endif /* HX711_H_ */
```

Prilog 3: Kod programa za upravljanje arduinom

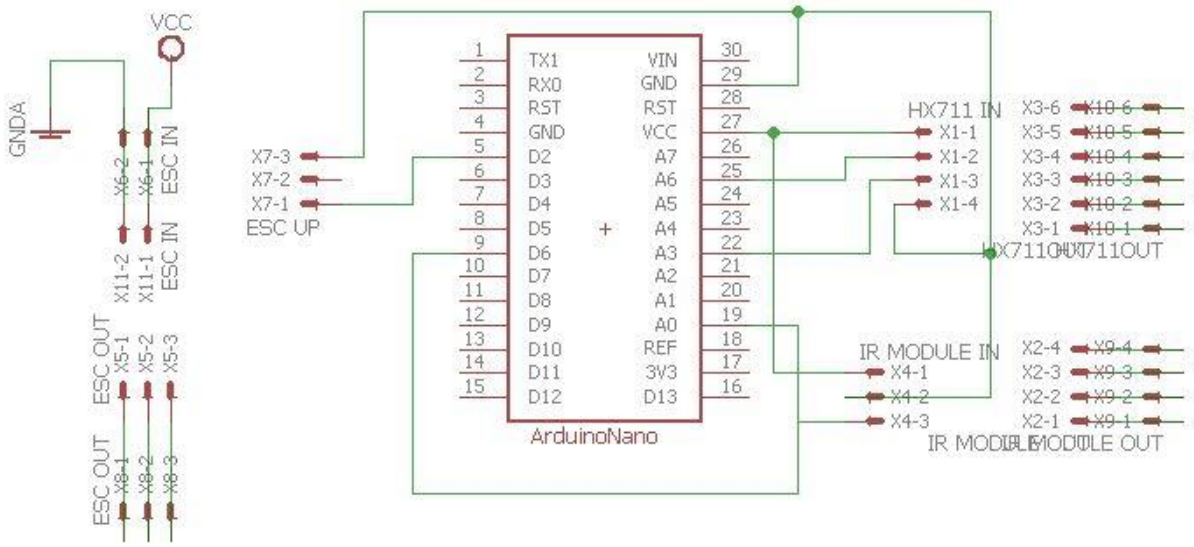
```
#include <Servo.h>
#include "Servo.h"
#include "hx711.h"

Hx711 scale(A1, A0);

Servo esc;

int escPin = 2 ;
int minPulseRate = 1000;
int maxPulseRate = 2000;
int throttleChangeDelay = 100;

void setup() {

  Serial.begin(9600);
  Serial.setTimeout(500);
  scale.setOffset(8623702);
  scale.setScale(169);
  // Attach the the servo to the correct pin and set the pulse range
  esc.attach(escPin, minPulseRate, maxPulseRate);
  // Write a minimum value (most ESCs require this correct startup)
  esc.write(0);

}
void loop() {
  // Wait for some input
  if (Serial.available() > 0) {

    // Read the new throttle value
    int throttle = normalizeThrottle( Serial.parseInt() );

    // Print it out
    Serial.print("Setting throttle to: ");
    Serial.println(throttle);
     // Change throttle to the new value
```

```
changeThrottle(throttle);
  }
}
void changeThrottle(int throttle) {
    // Read the current throttle value
  int currentThrottle = readThrottle();

  // Are we going up or down?
  int step = 1;
  if( throttle < currentThrottle )
    step = -1;

  // Slowly move to the new throttle value
  while( currentThrottle != throttle ) {
    esc.write(currentThrottle + step);
    currentThrottle = readThrottle();
    delay(throttleChangeDelay);
  }
}

int readThrottle() {
  int throttle = esc.read();

   Serial.print("Trenutni gas: ");
   Serial.println(throttle);
   Serial.print(" ");
   Serial.println(scale.getGram());


  return throttle;
}

// Ensure the throttle value is between 0 - 180
int normalizeThrottle(int value) {
  if( value < 29 )
    return 29;
  if( value > 150 )
    return 150;
  return value
```

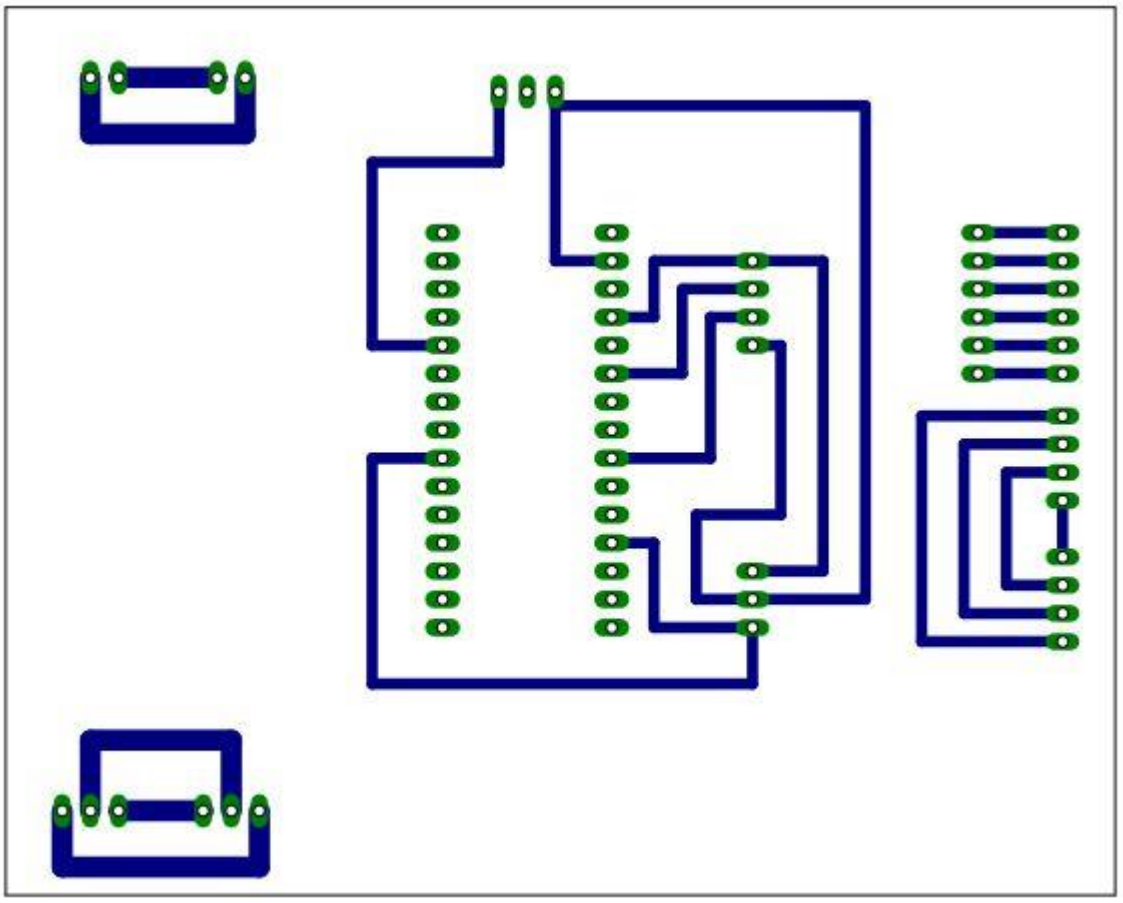
Prilog 4: Električna shema pločice u Eagle-u



Eagle board

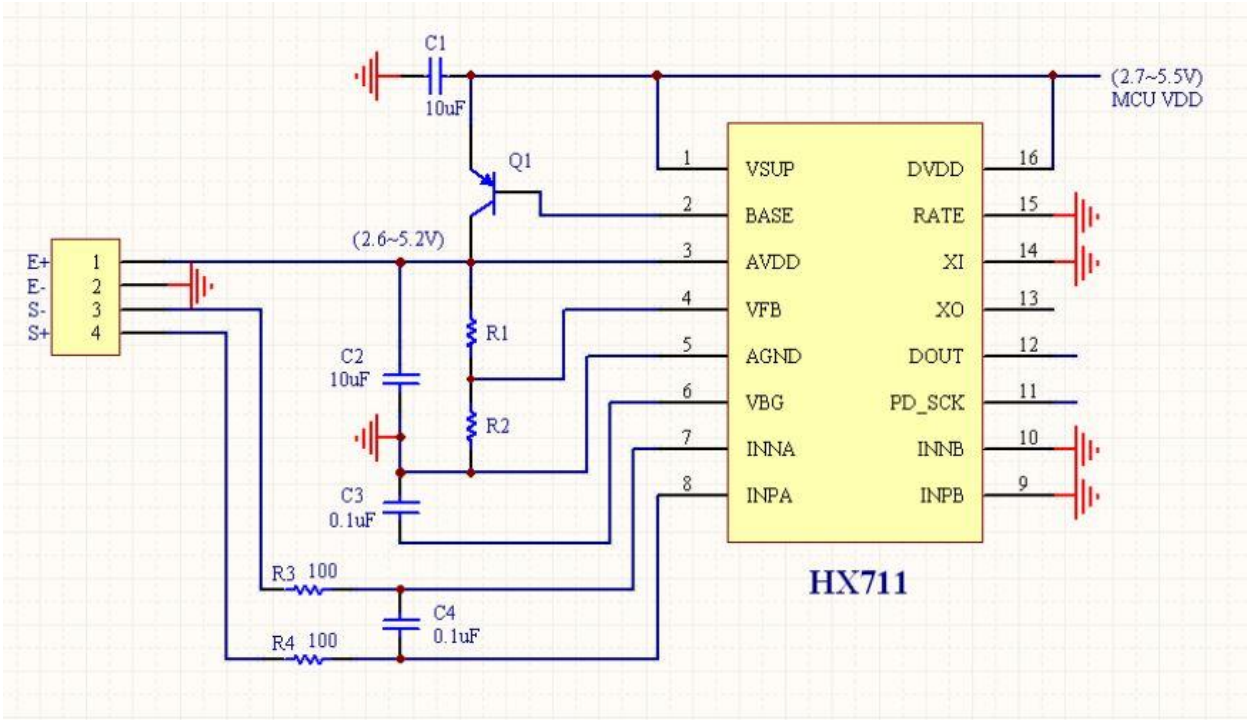


Eagle scheme



Eagle print

Prilog 5: Električna shema HX711 A/D pretvarača



Prilog 7. Program kod iz Visual Studia 2012

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Windows.Forms.DataVisualization.Charting;

namespace eugen_diplo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            foreach (string s in SerialPort.GetPortNames())
            {
                listBox1.Items.Add(s);
            }


            chart1.ChartAreas[0].AxisX.MinorGrid.Interval = 1;
            chart1.ChartAreas[0].AxisX.MinorGrid.Enabled = true;
            chart1.ChartAreas[0].AxisX.Title = "Vrijeme (ms)";
            chart1.ChartAreas[0].AxisY.Title = "Potisak (g)";
            chart1.ChartAreas[0].AxisX.MajorGrid.LineColor = Color.Gainsboro;
```

```
    chart1.ChartAreas[0].AxisY.MajorGrid.LineColor = Color.Gainsboro;
    chart1.ChartAreas[0].AxisX.MinorGrid.LineColor = Color.Gainsboro;
    chart1.ChartAreas[0].AxisY.MinorGrid.LineColor = Color.Gainsboro;
    button3.Enabled = false;
    if (timer1 != null)
    {
        timer1.Enabled = false;
    }
    radioButton5.Checked = true;


}
string brzina = "res";
SerialPort myPort;
string comPort = "COM4";
int baudRate=9600
int interval = 500;
int postotak=0;
string[] data = { "", "", "", "" };
string[] data1 = { "", "", "", "" };

bool opened = false;
private void button1_Click(object sender, EventArgs e
{

    if (opened == false) {
        comPort = listBox1.GetItemText(listBox1.SelectedItem);
    label12.Text = interval.ToString();
    InitTimer();
    button1.Text = "Port opened";
    button1.Enabled = false;
    button3.Enabled = true;
    opened = true;

}
    else
    {
        timer1.Stop();
        myPort.WriteLine("ug");
        myPort.Close();
        button1.Text = "Open port";
    }

}

private Timer timer1
public void InitTimer
{
    timer1 = new Timer();
    timer1.Tick += new EventHandler(timer1_Tick);

    timer1.Interval = interval;
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    myPort = new SerialPort(comPort, baudRate, Parity.None, 8, StopBits.One);

    myPort.Open();
    myPort.WriteLine(brzina);
    if (brzina == "res
    {
        postotak = klizac1.Value
        double umnozak = (postotak * 1.21) + 29
        brzina = Math.Round(umnozak).ToString
        label9.Text = postotak.ToString() + "%";
    }
```

```
            myPort.Close();

            myPort.DataReceived += new
SerialDataReceivedEventHandler(ReceivedSerialHandler
            myPort.Open();
        }
        private void ReceivedSerialHandler(object sender, SerialDataReceivedEventArgs e)
        {
            SerialPort sp = (SerialPort)sender

            this.Invoke((MethodInvoker)delegate
            {
                string recvData = sp.ReadLine();

                System.IO.StreamWriter file1 = new
System.IO.StreamWriter("C:\\save\\recvData.txt");
                file1.WriteLine(recvData);
                file1.Close();

                data = recvData.Split('/');
                int duljina = data.Length;

                if (duljina < 4)
                {
                    data = data1;
                }
                else
                {
                    data1 = data;
                    label7.Text = data[0
                    label8.Text = data[1
                    double broj1 = (Convert.ToInt32(data[2]) - 29) / 1.21
                    double broj2 = (Convert.ToInt32(data[3]) - 29) / 1.21;
                    string labela1=Math.Round(broj1).ToString() + "%";;
                    string labela2=Math.Round(broj2).ToString() + "%";;
                    label9.Text = labela1;
                    label14.Text = labela2;

                    System.IO.StreamWriter file2 = new
System.IO.StreamWriter("C:\\save\\data.txt");
                    file2.WriteLine(data[0]+";"+data[1]+";"+data[2]+";"+data[3]);
                    file2.Close();
                    System.IO.StreamWriter file3 = new
System.IO.StreamWriter("C:\\save\\data2.txt");
                    file3.WriteLine(data[0]+";"+data[1]+";"+labela1+";"+labela2);
                    file3.Close();
                }
                chart1.Series["Potisak"].Points.AddXY(data[0], data[1]);
            }
            );
            myPort.Close();
        }
        bool zaustavljen = false;
        private void button3_Click(object sender, EventArgs e)
        {
            if (zaustavljen == true)
            {
                timer1.Start
                button3.Text = "Zaustavi graf";
                zaustavljen = false;
            }
            else
            {
                timer1.Stop();
                button3.Text = "Pokreni graf";
                zaustavljen = true;
            }
```

```
}

private void button4_Click(object sender, EventArgs e)
{
    brzina = "res

}


private void textBox1_KeyDown_1(object sender, KeyEventArgs e
{
    if (e.KeyCode == Keys.Enter
    {
        button1_Click(this, new EventArgs());
    }
}

private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        button1_Click(this, new EventArgs());
    }
}

private void button5_Click(object sender, EventArgs e)
{
    chart1.Series[0].Points.Clear();
}

private void button6_Click(object sender, EventArgs e)
{

    InitTimer();
}

private void textBox4_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        button6_Click(this, new EventArgs());
    }
}

private void button7_Click(object sender, EventArgs e)
{
    chart1.Series[0].Points.Clear();
    brzina = "res";
}

private void radioButton5_CheckedChanged(object sender, EventArgs e)
{
    if (opened == true)
    {
        timer1.Stop();
        interval = 500;
        label12.Text = interval.ToString();
        InitTimer();
    }
}

private void radioButton4_CheckedChanged(object sender, EventArgs e)
{
    if (opened == true)
    {
        timer1.Stop();
        interval = 400;
        label12.Text = interval.ToString();
```

```
        InitTimer();
    }
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
   if (opened == true)
    {
        timer1.Stop();
        interval = 300;
        label12.Text = interval.ToString();
        InitTimer();
    }
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (opened == true)
    {
        timer1.Stop();
        interval = 250;
        label12.Text = interval.ToString();
        InitTimer();
    }

}

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (opened == true)
    {
        timer1.Stop();
        interval = 200;
        label12.Text = interval.ToString();
        InitTimer();
    }
}

private void button8_Click(object sender, EventArgs e)
{
    brzina = "ug";
    klizac1.Value = 0;
    postotak = klizac1.Value;
    double umnozak = (postotak * 1.21) + 29
    label9.Text = "STOPPED";
    label3.Text = "Brzina: STOPPED";

}

private void klizac1_Scroll(object sender, ScrollEventArgs e)
{
    label15.Text = (klizac1.Value).ToString();
}

private void klizac1_MouseLeave(object sender, EventArgs e)
{
    postotak = klizac1.Value;
    label3.Text = "Brzina: " + postotak.ToString() + "%";
    double umnozak = (postotak * 1.21) + 29;
    brzina = Math.Round(umnozak).ToString();
    label9.Text = postotak.ToString() + "%";
}}}
```