SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ELEKTROTEHNIČKI FAKULTET

Diplomski studij procesnog računarstva

AUTOMATIZIRANO NJIHALO

Elementi automatike

Bruno Jaman

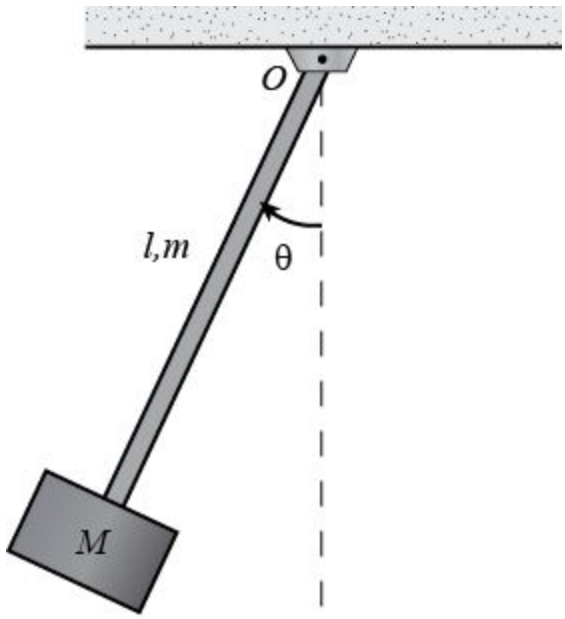
Josip Krušec

Osijek, 2016.

SADRŽAJ

1. UVOD

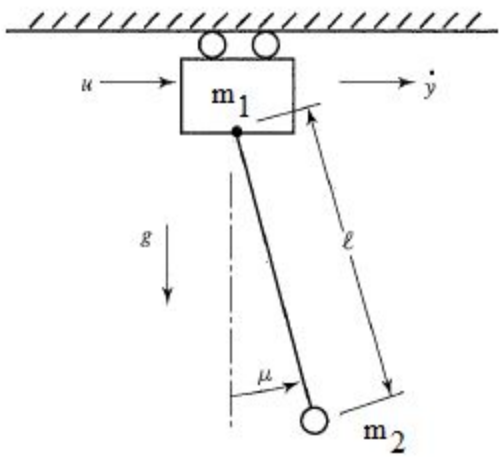
Projektni zadatak ovoga seminara je projektirati i realziriti automatizirano njihalo. Njihalo podrazumijeva štap učvšćen u točki podloge tako da se štap može otklanjati za određeni kut lijevo ili desno od normale na tu točku. Na suprotnom kraju se nalazi tijelo mase M.[1]



Slika 1. Primjer njihala [2]

Realizirano njihalo je automatsko iz razloga što je točka O pomična i može se programski upravljati njezinom pozicijom koristeći koračni motor. Nakon što točka O prijeđe određeni put, unesen od strane korisnika, o stabilizaciji njihala brine PID regulator, što je još jedan razlog koji čini ovo njihalo automatskim.

Na raspolaganje nam je stavljen pisač, koji smo okrenuli naopako i postavili ga na nosač. Na glavu printera postavili smo štap i na štap stavili sklop koji u sebi sadrži žiroskop i akcelerometar kao masu m2. Masa m1 predstavlja masu glave printera koja je na slici 2 prikazana kao kolica. Bitne veličine su : sila kojom se djeluje na kolica(u našem slučaju tu silu proizvodi motor),y' brzina kolica, l duljina štapa, μ kut otklona njihala, g akceleracija zemljine sile teže.



Slika 2. Fizikalni model automatiziranog njihala

Njihala su predmet proučavanja fizičara dugo vremena, još od Galilea i svoju široku pirmjenu nalaze i danas.[2]

Jedna od najčešćih primjena je u satovima, gdje se njihalo koristi za kontrolu gibanja kazaljki. Kada se njihalo pomakne naprijed i nazad pomiče zupčanik za jedan usjek. Zupčanici zatim pomiču kazaljke sata. [2]

Foucaltovo njihalo je jednostavan uređaj koji se koristi kao dokaz za Zemljino rotacijsko gibanje. Njihalo je realizirano sa velikom duljinom štapa, iz razloga što njihala sa manjom duljinom l imaju tendenciju zaustavljanja pod utjecajem sile trenja. Foucalt je otkrio da kada se opisano njihalo njiše duži period tijekom dana, čini se kao da njihalo mijenja smjer. Ono što se zapravo događa je da njihalo stalno ima isti smjer, ali Zemlja se rotacijski giba u odnosu na njihalo.[3]

Slika 3. Foucaltovo njihalo Pantheon, Paris[3]

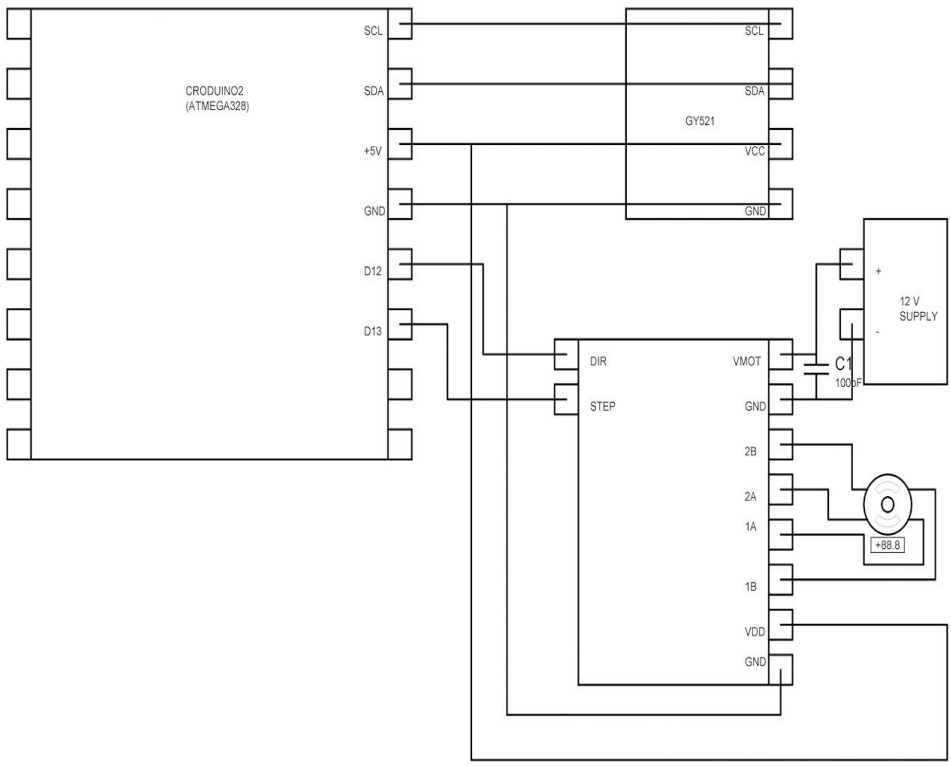
Njihalo je našlo svoju primjenu u seizmografiji. Radi se o malo drugačijoj izvedbi njihala, no što je gore navedeno. Čvrsta točka O se može gibati dok je masa na kraju štapa

učvršćena i štap je postavljen vodravno. Kada se dogodi određen Zemljin tremor ili potres, točka O se pomiče, dok masa m na kraju štapa ostaje na istom mjestu i razlika je intenzitet podrhtavanja.

U moderno vrijeme, u automatici, automatizirana njihala se koriste kao okosnica kranova koji prenose i točno pozicioniraju težak teret.[4] Također, automatizirana njihala koriste se za nadziranje pokreta horizontalnih struktura kao što su: mostovi, brane i ostalih visokih građevine.[5]

2. PROJEKTIRANJE SUSTAVA

Sustav je zamišljen tako da postoji šina koja predstavlja putanju po kojoj se kolica, na koja je postavljeno njihalo, mogu gibati u jednoj dimenziji tj. lijevo i desno. Kolica su prijenosom povezana na koračni motor koji kontrolira smjer i brzinu gibanja kolica, samim time i poziciju njihala. Ideja je dodati i driver za koračni motor kako bi se omogućila lakša kontrola koračnog motora. Na kraj njihala postavljen je senzor sa žiroskopom i akcelerometrom. Senzor je zamišljen kao element povratne veze, kako bi se mogao implementirati PID regulator koji kontrolira stabilizaciju njihala nakon prijeđene zadane udaljenosti. Svi elementi spojeni su na croduino pločicu kako bi se omogućilo sučelje za unos pozicije njihala te kako bi se omogućila povezanost i smisleno kontroliranje svih elemenata u sustavu.



Slika 4. Električna shema sustava

2.1. Korištene komponente

2.1.1. Pisač i koračni motor minebea 17PM-H103-09V

Na raspolaganje nam je stavljen stari lexmark inkjet pisač koji u sebi sadrži koračni motor minebea 17PM-H103-09V. Pisač je zatečen u neispravnom stanju, ali koračni motor radi ispravno. Sa pisača smo uklonili nepotrebne dijelove, a iskoristili smo šinu po kojoj se kreće glava pisača i koračni motor. Šina je modelirana na način da nam predstavlja putanju po kojoj će se kretati glava motora koja predstavlja pomićna kolica, na koja smo pričvrstili njihalo.

Minebea 17PM-H103-09V je unipolarni koračni motor sa četiri izvoda sa zavojnica, tj. dva izvoda za napajanje svake od zavojnica. Broj stupnjeva po koraku iznosi 1.8, nominalni napon za napajanje zavojnica iznosi od 5 do 12 V istosmjernog napona, konzumacija struje iznosi 1.2 A po fazi, moment je 1 kgcm, masa iznosi 273 g.

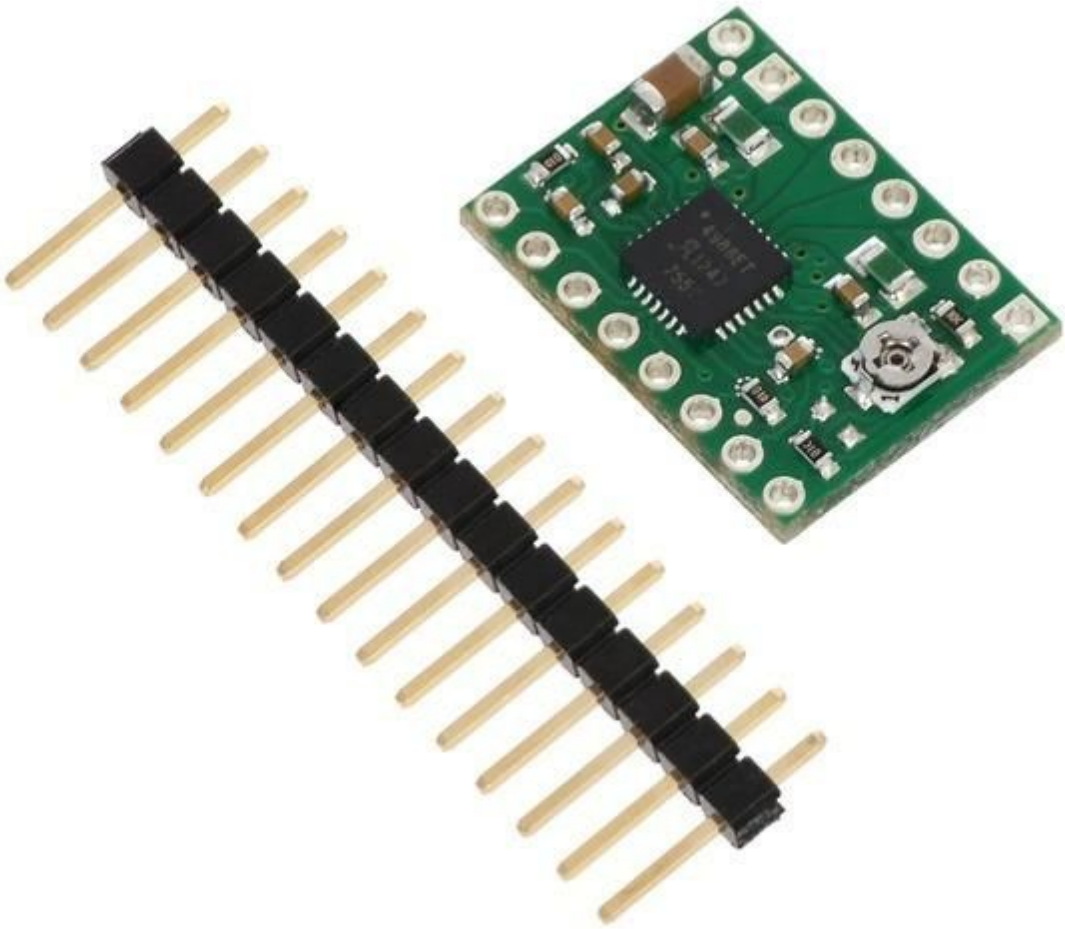
Slika 5. Izgled Minebea 17PM-H103-09V koračnog motora[6]

2.1.2. Driver za koračni motor A4988

A4988 pruža jednostavno sučelje za upravljanje koračnim motorom. Karakteristike koje posjeduje su :

- sučelje za upravljanje motorom preko dvije veličine - broja koraka i smjer gibanja
- pet razlučivosti koraka : puni, polovina, četvrtina, osmina i šesnaestina koraka
- postavljanje maksimalne izlazne struje preko ugrađenog potenciometra
- zaštita od pregrijavanja, zaključavanje uslijed nedovoljnog napona i zaštitu od krosovera struje

- zaštita od kratkog spoja[7]



Slika 6. Izgled drivera za koračni motor Pololu A4988 [8]

2.1.3. Senzor GY-521 (MPU 6050)

MPU 6050 sadrži 3-osni žiroskop i 3-osni akcelerometar, što zajedno daje 6 stupnjeva slobode. I žiroskop i akcelerometar se mogu koristiti zasebno, no najčešće se kombiniraju kako bi se eliminirale pogleške u mjerenju. Žiroskop ima tendenciju drifta, dok je akcelerometar osjetljiv na šum. Njihovim kombiniranjem postiže se sinergija. [9]

Karakteristike :

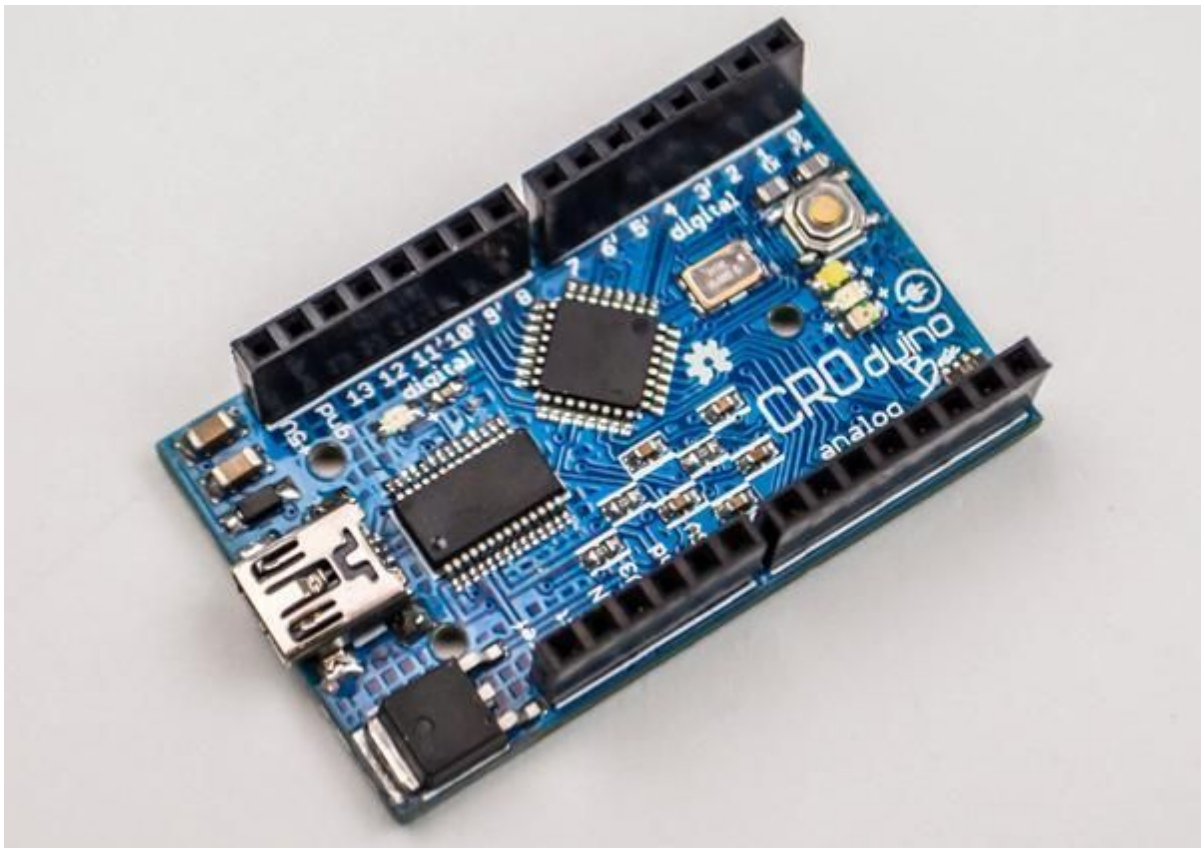
- raspon akcelerometra : ±2, ±4, ±8, ±16g
- raspon žiroskopa : ± 250, 500, 1000, 2000 °/s
- operativni napon: 3.3 V- 5V [10]

Slika 7. Izgle senzora MPU 6050 [11]

2.1.4. Croduino Basic 2 pločica

Croduino Basic 2 je nova generacija hrvatskih pločica kompatibilnih sa Arduinom. Za komunikaciju sa računalom zadužen je USB pretvarač CP2102. Od kratkog spoja ga štiti silicijski osigurač. 100% je kompatibilan sa Arduino Nano. Sadrži mikrokontroler Atmel Atmega328. Karakteristike su slijedeće :

- Atmel Atmega328 mikrokontroler(32kB flash, 2kB RAM, 0.5 kB EEPROM)
- Silabs CP2102 USB to UART most
- 22 ulazno-izlazna pina, 14 digitalnih ulazno-izlaznih pinova, 8 analognih ulaza, 6 PWM-a
- Veličina 3 x 5 cm
- LED dioda na 13-om pinu
- Regulator za mogućnost vanjskog napajanja
- Napravljen u Hrvatskoj [12]



Slika 8. Izgled Croduino Basic 2 pločice [13]

2.1.5. Napajanje Linkworld LPJ12-25-P4 420W

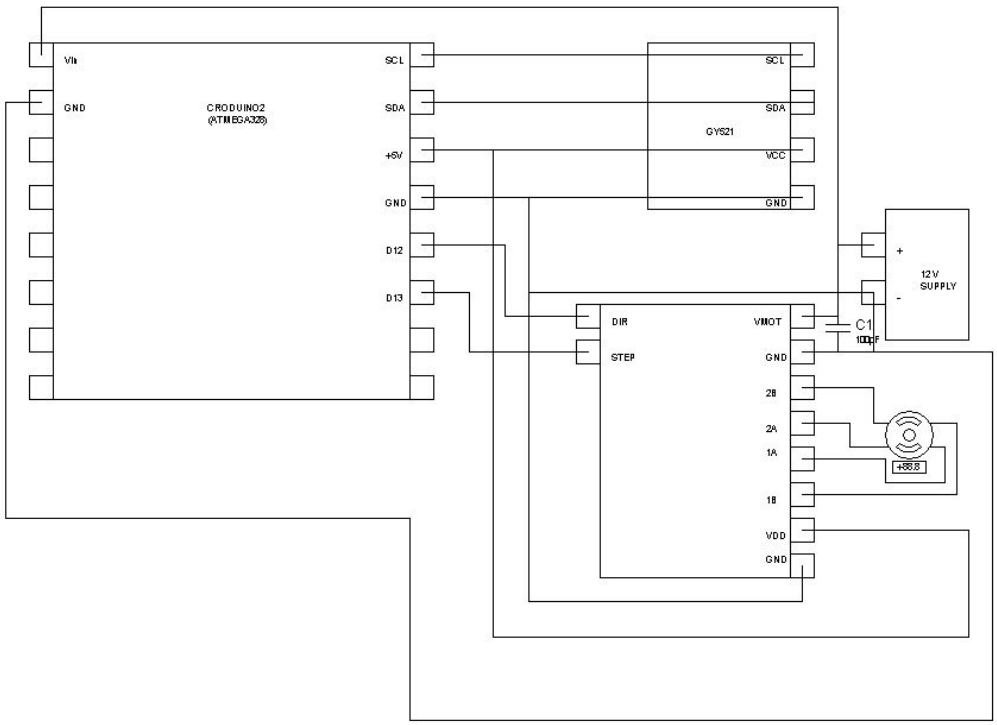
Za napajanje sustava iskoristili smo napajanje izvađeno iz računala. Karateristike su slijedeće:

- Maksimalna snaga 420 W
- Glavni kontektor 20+4 pin
- Dimenzije 3.5''x5.5''x5.75''
- 1 konektor 12 V(P4)
- 2 periferiska konektora
- 2 SATA konektora [14]

Slika 9. Izgled napajanja sustava [15]

3. REALIZACIJA SUSTAVA

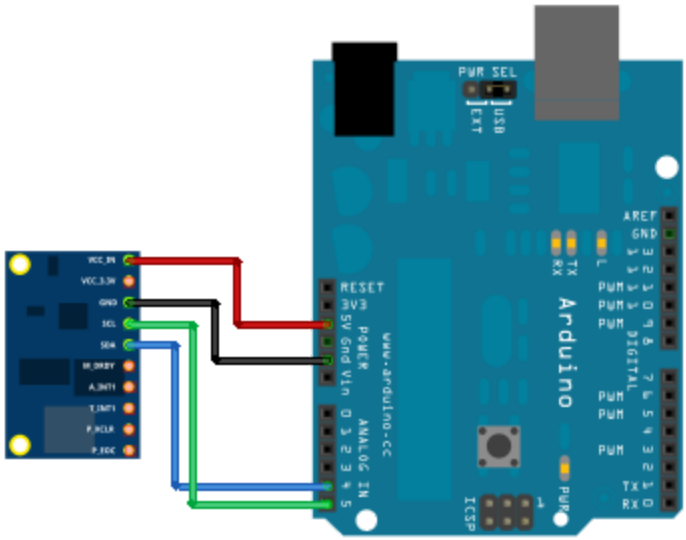
Sustav je realiziran na naičin kako je i projektiran prema shemi sa slike 4, sa sitnim izmjenama. Zbog problema sa komunikacijom između senzora MPU 6050 i croduina, cijeli sustav smo spojili na zajednički izvor od 12 V i na zajedničko uzemljenje. Sklopovi koji rade na 5V napajani su preko izvora od 5V sa croduino pločice. Shema realiziranog sustava, nakon promjena izgleda kao što je prikazano na slici 10.



Slika 10. Shema realiziranog sustava

3.1. Spajanje senzora MPU 6050 sa croduinom

MPU 6050 ima 4 pina, od kojih 2 za napajanje GND i 5V te 2 za komunikaciju SDA i SCL.Npajanje je izvedeno preko croduina na način da smo GND od senzora spojili sa GND od croduina te SDA i SCL sa odgovarajućim SDA i SCL pinovima na Croduinu. Spajanje je izvedeno prema shemi sa slike 11.



Slika 11. Električna shema spajanja MPU 6050 sa Aruinom[16]

Komunikacija između Croduina i MPU 6050 se odvija preko I2C protokola. Standardne funkcije za citanje i pisanje sa sklopa povezanog sa Croduinom preko I2C protokola pruža biblioteka Wire.h. Funkcija za čitanje podataka sa senzora je i2cRead(), dok je funkcija za slanje podataka na senzor i2cWrite(). Programski komunikacija je izvedena prema kodu iz tablice 1.

```
void setup() {
  Serial.begin(115200);
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to all four registers at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis gyroscope reference and disable sleep mode

  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
    Serial.print(F("Error reading sensor"));
    while (1);
  }

  delay(3000); // Stabilizacija senzora

  /* Set kalman and gyro starting angle */
  while (i2cRead(0x3B, i2cData, 6));
  accX = (i2cData[0] << 8) | i2cData[1];
  accY = (i2cData[2] << 8) | i2cData[3];
  accZ = (i2cData[4] << 8) | i2cData[5];

  // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and eq. 26
  // atan2 outputs the value of -π to π (radians) - see http://en.wikipedia.org/wiki/Atan2
```

```
  // It is then converted from radians to degrees
#ifdef RESTRICT_PITCH // Eq. 25 and 26
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll); // Set starting angle
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;

  timer = micros();
}

void measure () {
  /* Update all the values */
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
  accZ = ((i2cData[4] << 8) | i2cData[5]);
  tempRaw = (i2cData[6] << 8) | i2cData[7];
  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];

  double dt = (double)(micros() - timer) / 1000000; // Delta vrijeme -  vrijeme jednog ciklusa
loopa u sekundama
  timer = micros();

  // Source: http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf eq. 25 and
eq. 26
  // atan2 outputs the value of -π to π (radians) - see http://en.wikipedia.org/wiki/Atan2
  // It is then converted from radians to degrees
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;   //konvertiranje iz radijana u
stupnjeve

  double gyroXrate = gyroX / 131.0; // gyroXrate - konveriranje u deg/s
  double gyroYrate = gyroY / 131.0; // gryoYrate - konvertiranje u deg/s

  kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); // kut X s Kalmanovim filterom

  // Ovo popravlja problem tranzicije kada kut akcelerometra skoci od -180 do 180
  if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90)) {
    kalmanY.setAngle(pitch);
```

```
    compAngleY = pitch;
    kalAngleY = pitch;
    gyroYangle = pitch;
  } else
    kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt); // kut Y s Kalmanovim filterom

  if (abs(kalAngleY) > 90) {
    gyroXrate = -gyroXrate; // Invert rate, so it fits the restriced accelerometer reading
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
    kalAngleX = kalAngleX - 1.2;
  }// Calculate the angle using a Kalman filter

  gyroXangle += gyroXrate * dt; // gyro angle bez filtera
  gyroYangle += gyroYrate * dt;
  //gyroXangle += kalmanX.getRate() * dt; // Calculate gyro angle using the unbiased rate
  //gyroYangle += kalmanY.getRate() * dt;

  /* Print Data */
  /*#if 0 // Set to 1 to activate
    Serial.print(accX); Serial.print("\t");
    Serial.print(accY); Serial.print("\t");
    Serial.print(accZ); Serial.print("\t");

    Serial.print(gyroX); Serial.print("\t");
    Serial.print(gyroY); Serial.print("\t");
    Serial.print(gyroZ); Serial.print("\t");

    Serial.print("\t");
    #endif */

  //Serial.print(roll); Serial.print("\t");
  //Serial.print(gyroXangle); Serial.print("\t");
  //Serial.print(compAngleX); Serial.print("\t");
  Serial.print("\t"); Serial.print(kalAngleX); Serial.print("\n"); //s ovime treba raditi

  //Serial.print("\t");

  //Serial.print(pitch); Serial.print("\t");
  //Serial.print(gyroYangle); Serial.print("\t");
  //Serial.print(compAngleY); Serial.print("\t");
  //Serial.print(kalAngleY); Serial.print("\t"); //s ovime treba raditi

  //Serial.print("\r\n");

}
```

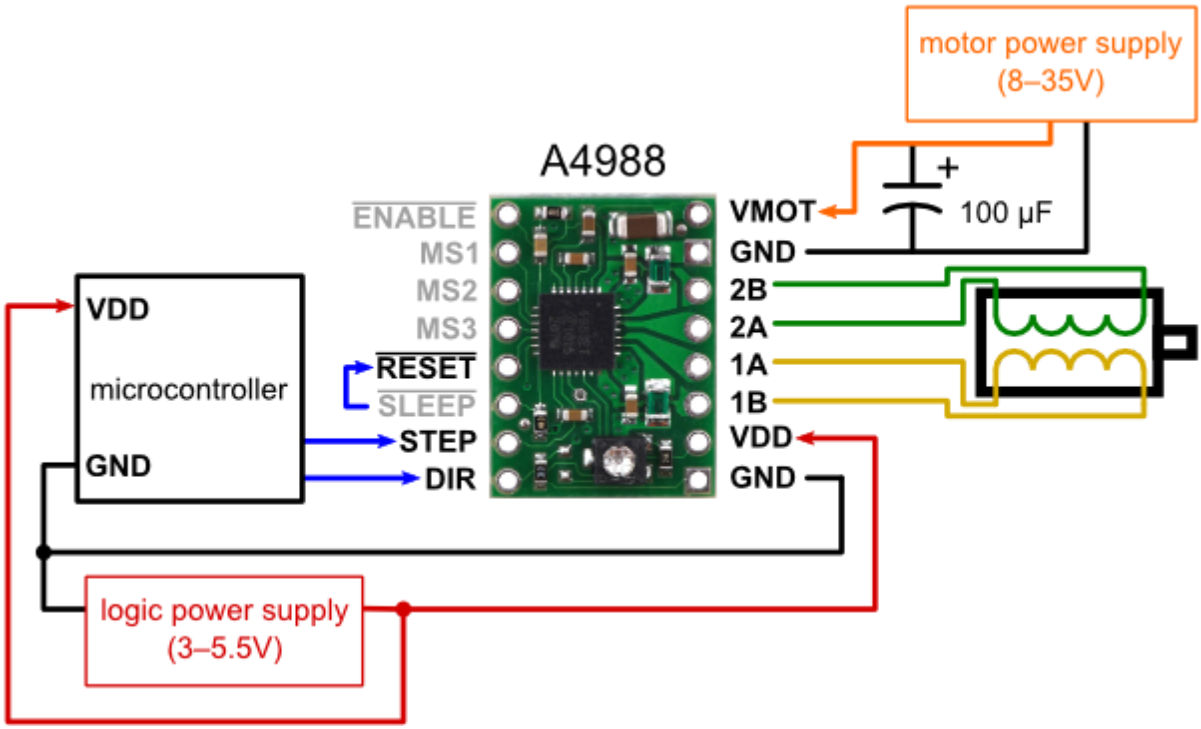
Tablica 1. Programska realzicaija čitanja podataka sa MPU 6050 s Kallmanovim filtrom

Kod je preuzet sa službene stranice arduino zajednice. Pošto je navedeni kod izdan pod licencom otvorenog koda, prema kojoj je dopusteno modificiranje koda i korištenje za osobnu upotrebu i učenje, nismo prekršpili nikakve odrednice licence.

U kod je ubačen Kallmanov filter pomoću kojega se vrši više mjerenja i estimacije nepoznatih varijabli, sve u svrsi dobivanja preciznijih podataka.

3.2. Spajanje drivera A4988 s croduinom i koračnim motorom

Kako bi sučelje prema motoru, preko drivera bilo omogućeno, sklopovski je potrebno spajanje izvršiti prema shemi sa slike 12.



Slika 12. Električna shema spoja drivera s motorom i croduinom [17]

Pinove STEP i DIR sa motora potrebno je spojiti sa digitalnim pinovima croduina. U našem slučaju to su D12 sa DIR pinom te D13 sa STEP pinom. Na ovaj način je realizirana kontrola motora preko dvije varijable. STEP koja predstavlja broj koraka motora i DIR koja predstavlja smjer gibanja motora. Pinovi RESET i SLEEP su kratko spojeni. Pinovi VDD i GND spojeni su na 5V i GND croduina. Pinovi 1A i 1B spojeni su na izvode jedne zavojnice motora, dok su pinovi 2A i 2B spojeni na izvode druge zavojnice motora. Pinovi VOMT i GND su spojeni sa vanjskim napajanjem od 12V preko kondenzatora od 100 µF.

3.4. Upravljanje motorom

Projektni zadatak je realizirati unos duljine pomaka koju unosi korisnik te stabilizaciju njihala nakon pređenog pomaka. Za stabilizaciju njihala zadužen je PID regulator i njegova realizacija biti će objašnjena u slijedećem potpoglavlju. U ovom potpoglavlju ćemo objasniti sučelje za unos pomaka i upravljanje motorom na temelju unesene veličine.

```
if (Serial.available() > 0) {   //unosenje duljine pomaka u centimetrima
  lenght = Serial.parseInt();
  Serial.println(lenght);
  myStepper.setSpeed(motorSpeed);
  lenghtcm = lenght * 130;
  myStepper.step(lenghtcm);
  left = 0;
  right = 0;
}
```

Tablica 2. Programska realizacija unosa duljine pomaka preko serijskog monitora

Kod sa tablice 2. instruira mikrokontroler da ako postoji unos na serijskom monitoru da ga parsira u cijeli broj i spremi u varijablu lenght. Pošto je varijabla lenght u cm, a nama su potrebni koraci motora, proračunom smo došli da je potrebno 130 koraka motora da bi se presao 1 cm. Iz toga proizlazi da varijabla lenght pomnožena sa brojem 130 daje broj koraka motora ekvivalentnih duljini u centimetrima. Metodom setSpeed klase myStepper postavlja se brzina motora na predefiniranu vrijednost, do koje smo emprijiski došli da je najviša.

3.5. Realizacija PID regulatora

PID regulator radi zadanom frekvencijom od 50 Hz. Dopušteno odstupanje od jednog stupnja je stavljeno jer sustav ne treba stalno regulirati. Micanjem toga uvjeta može se vidjeti kako sustav reagira kada ga nije moguće dovesti u željenu točku - njihalo konstantno titra.

Ulazna veličina regulatora je otklon njihala mjeren senzorom i filtriran Kallmanovim filtrom a izlaz brzina pomaka motora pri reguliranju. U realizaciji se koristi adaptivno upravljanje. Regulator koristi različita pojačanja s obzirom na udaljenost sustava od željene točke, za greške manje od 15 koriste se konzervativni parametri.

Proporcionalno djelovanje računa se množenjem greške (*error)* sa zadanim parametrom pojačanja.

Integralno djelovanje računa se kao zbroj svih grešaka sustava tijekom djelovanja regulatora. Kod integralnog djelovanja pojavljuje se problem *windup*-a. Kako je integralno djelovanje zbroj svih prijašnjih grešaka, ako sustav počne s radom vrlo daleko od željene točke zbroj grešaka može postati prevelik i integralni dio može dati dominantni efekt koji sprječava brzi dolazak u željenu točku. Ovaj problem riješen je "*zero out"* metodom koji postavlja integralni dio regulatora na nulu ako greška sustava prelazi zadanu graničnu vrijednost, u ovom sustavu to je greška veća od 20 stupnjeva. Ovaj uvjet dopušta integralnom djelovanju da radi samo kada je sustav već blizu željene točke, i onda ono nastoji ukloniti male greške i postaviti sustav u ravnotežu. [18]

Derivativno djelovanje računa se kao umnožak derivativnog pojačanja kD i razlike pozicije sustava u trenutnoj i prethodnoj iteraciji djelovanja regulatora.

Nakon izračuna PID djelovanja, vrijednosti se mapiraju u interval do 170 i nakon toga ograniči na maksimalnu vrijednost od 170 jer mapiranje može dati i veću vrijednost od toga u slučaju da regulatora prelaze pretpostavljenu gornju granicu. 170 je najveća brzina motora pri kojoj on još uvijek ispravno i konstantno radi.

Posljednji korak je određivanje smjera kretanja motora s obzirom na predznak kuta i pokretanje motora u zadanom smjeru za određeni put reguliranom brzinom.

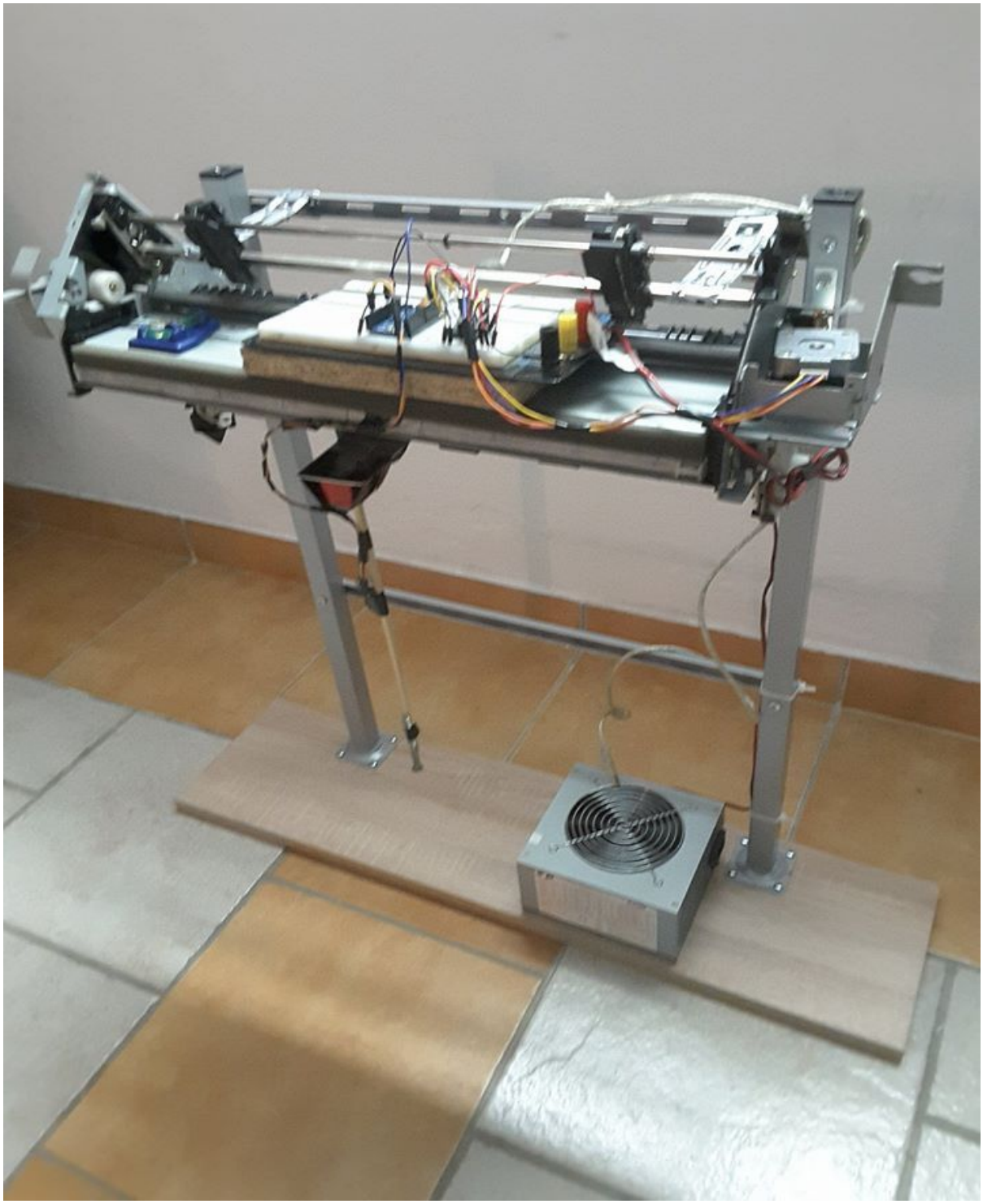
```
int kutni_pid () {

  t = millis();
  pidtimer = (double)(t - lasttime);
  if (pidtimer >= sampletime) {    //osigurava frekvenciju reguliranja 50hz

    actual = abs(kalAngleX);  //kut X bez predznaka
    error = 0 - actual;     //greska

    if (error > -1) {  //dopusteno odstupanje
      integral = 0;
      return 0;
    }

    if (error > intTresh) { //prevencija integralnog windupa, treshold = 45 stupnjeva
      integral = integral + error;
    }
    else integral = 0;

    if ( error > 15) {
      P = error * kP;
```

```
      I = integral * kI;
      D = (last - actual) * kD;
    }
    else {                //ako smo blize zadanoj vrijednosti - koristi konzervativne parametre
regulatora
      P = error * kPcons;
      I = integral * kIcons;
      D = (last - actual) * kDcons;
    }
    drive = abs(P + I + D);
    drive = doubleMap(drive, 0, 300, 50, 170);  //mapiranje izlaza
    drive = constrain(drive, 0, 170);  // ogranicavanje izlaza da ne prelazi maksimalnu
brzinu

    myStepper.setSpeed(drive);
    if (kalAngleX < 0) {     // smjer kretanja
      myStepper.step(20); // duljina kretanja
    }
    else
      myStepper.step(-20);

    last = actual;  // kut za koristenje u iducoj iteraciji
    lasttime = millis();
  }
  //else Serial.print ("nije vrijeme");
}
```

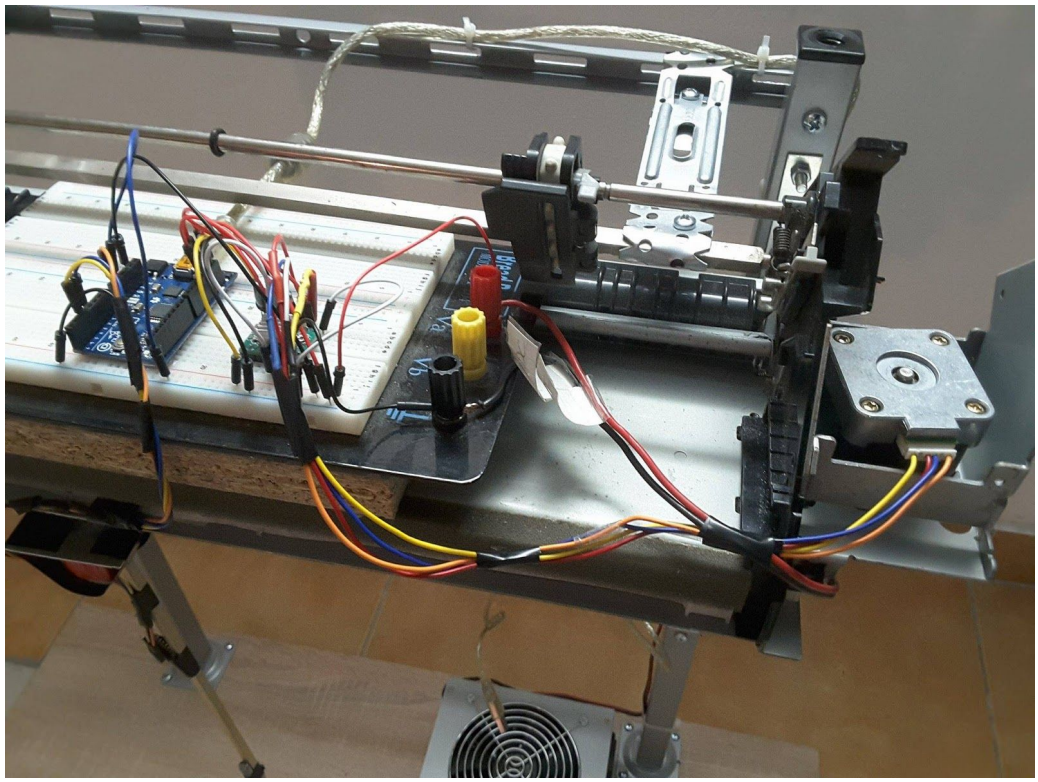
Tablica 3. Programska implementacija PID regulatora

3.6. Izgled sustava

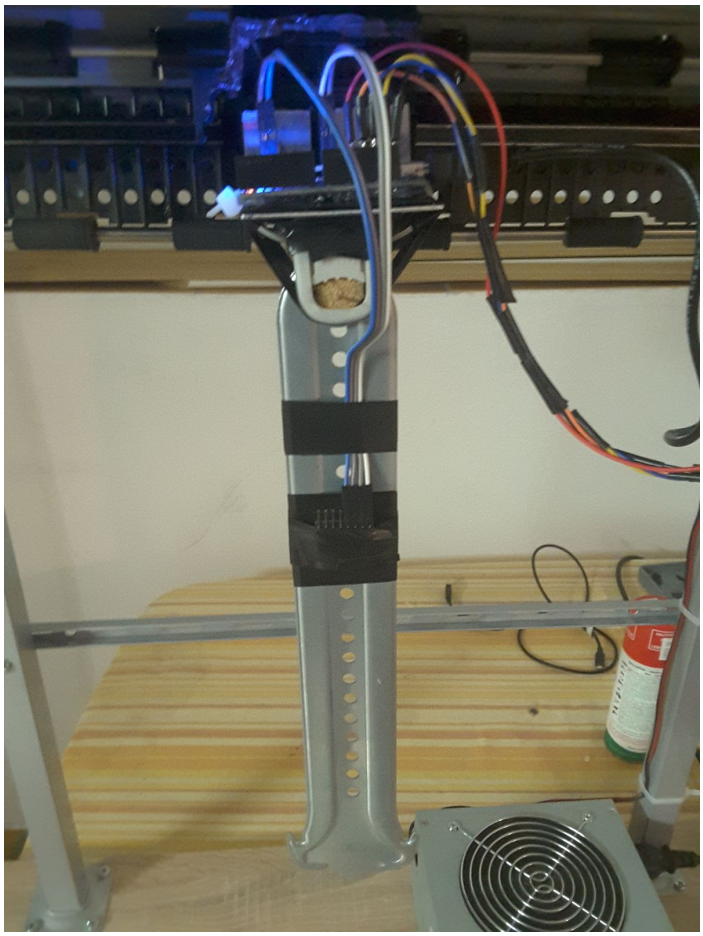
Nakon realizacije opisane u prethodnim podpoglavljima, sustav poprima izgleda sa slika 13 i 14.



Slika 13. Izgled realiziranog sustava



Slika 14. Izgled realiziranog spoja komponenata na protoboardu

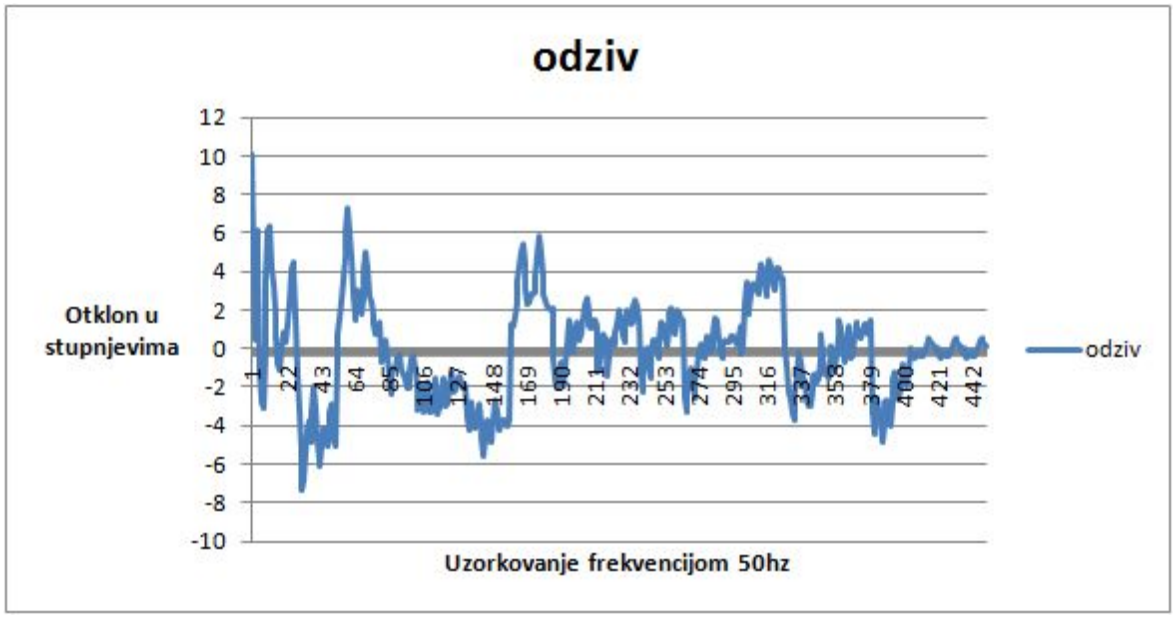


Slika 15. Druga verzija njihala

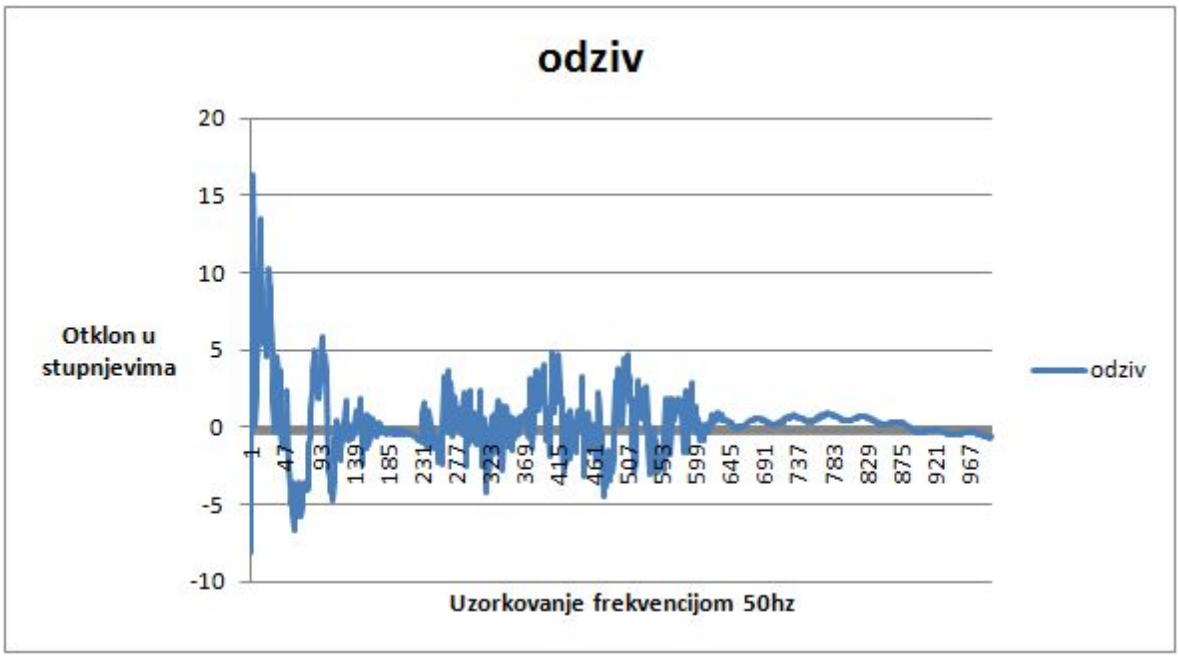
4. TESTIRANJE

Parametri regulatora 1: kP = 30, kI = 1, kD = 30, kPcons = 20, kDcons = 0.5, kIcons = 20

.

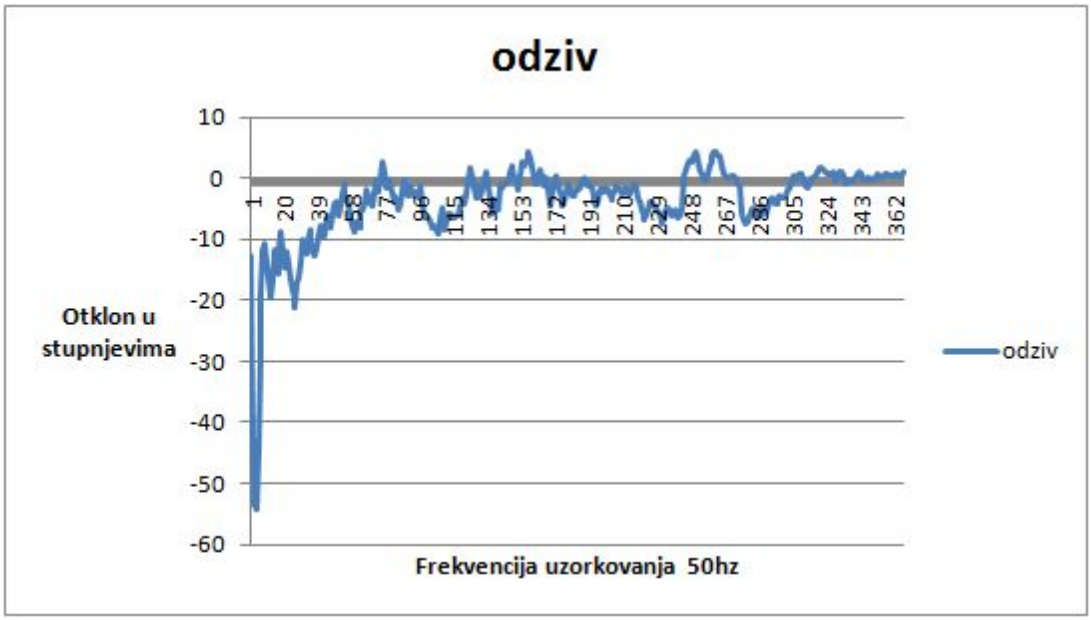


Slika 16. Odziv sustava na pomak 5cm u lijevo s parametrima 1

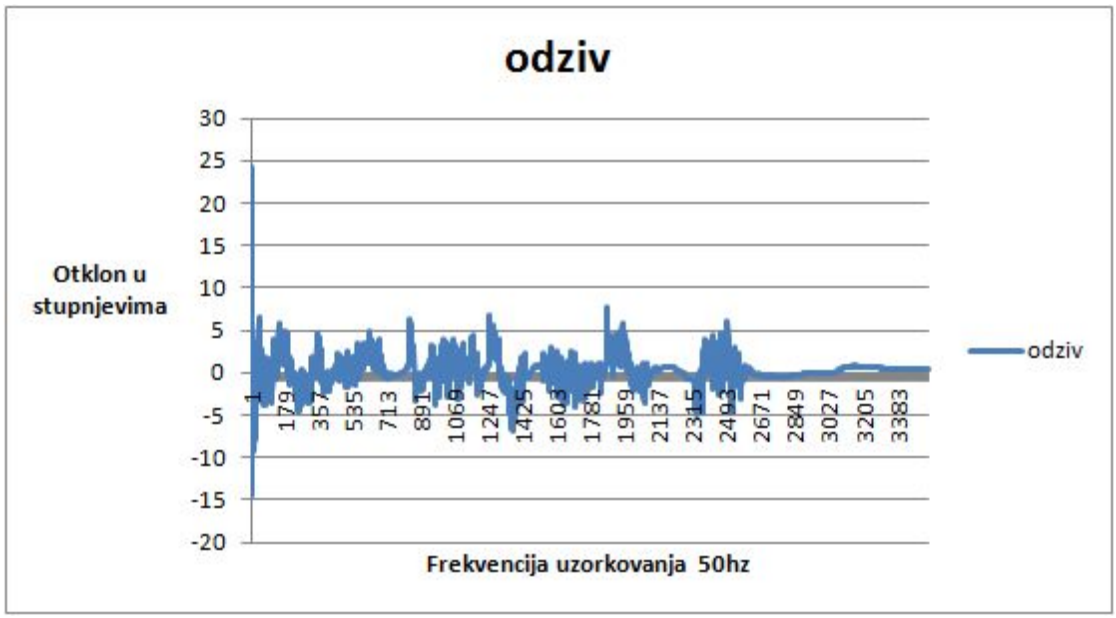


Slika 17. Odziv sustava nakon pomaka 15cm u desno s parametrima 1

Parametri regulatora 2: kP = 20, kI = 0.5, kD = 20, kPcons = 16, kDcons = 0.3, kIcons = 16

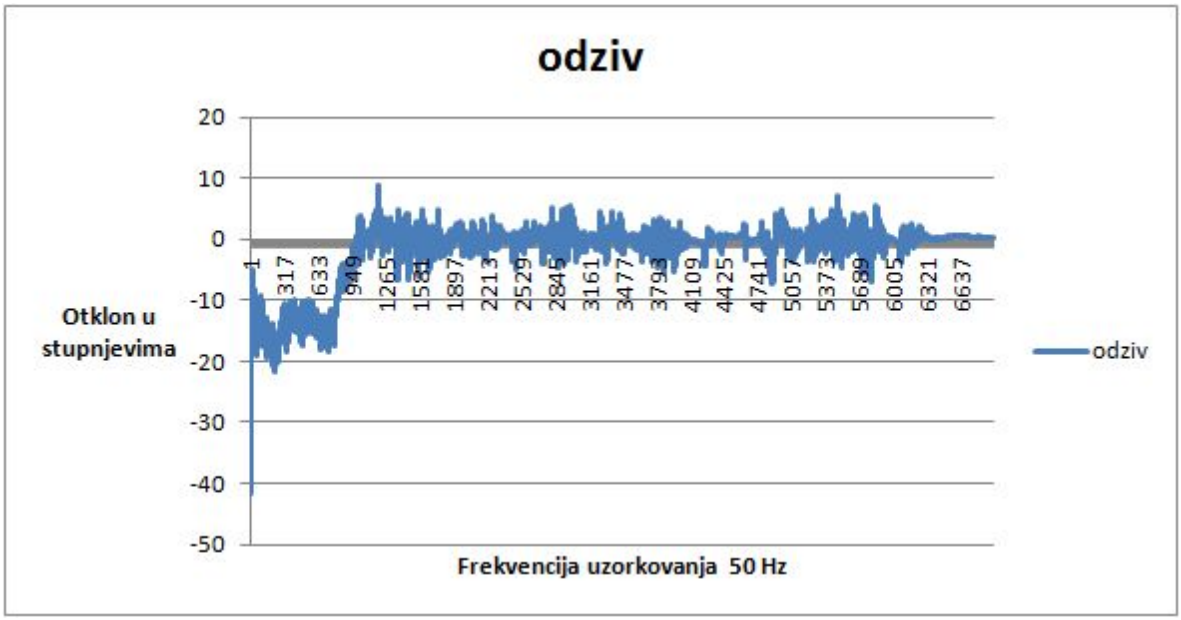


Slika 18. Odziv sustava nakon pomaka 10cm u lijevo s parametrima 2

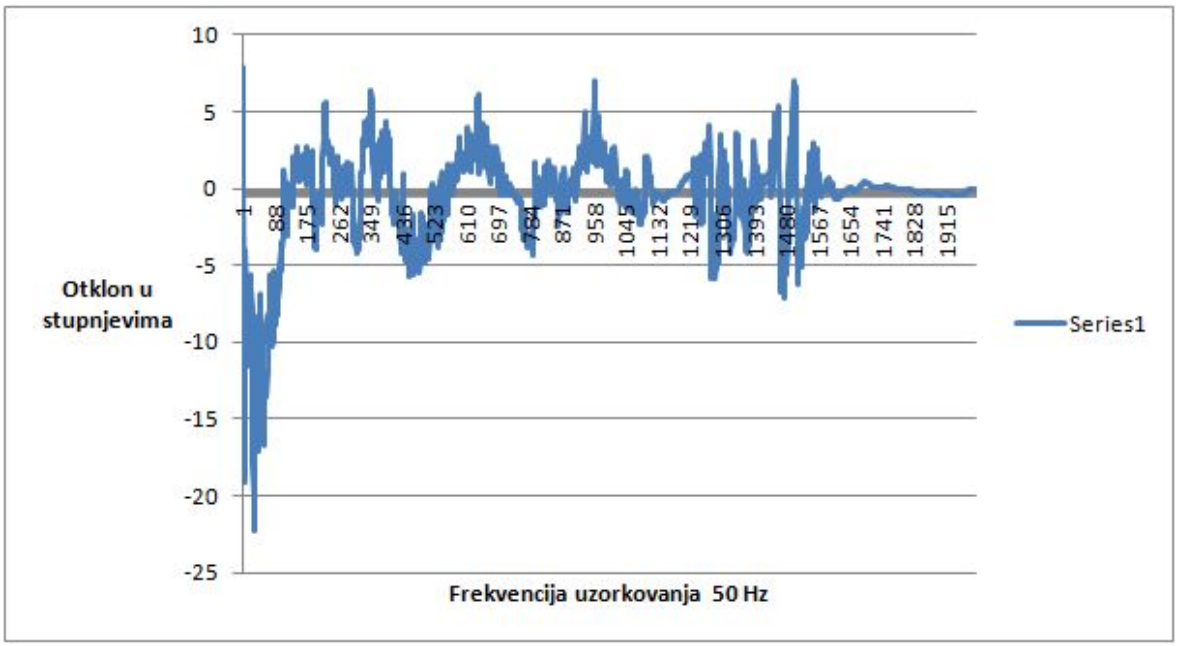


Slika 19. Odziv sustava nakon pomaka 5cm u desno s parametrima 2

Parametri regulatora 3: kP = 40, kI = 1, kD = 40, kPcons = 25, kDcons = 0.7, kIcons = 32

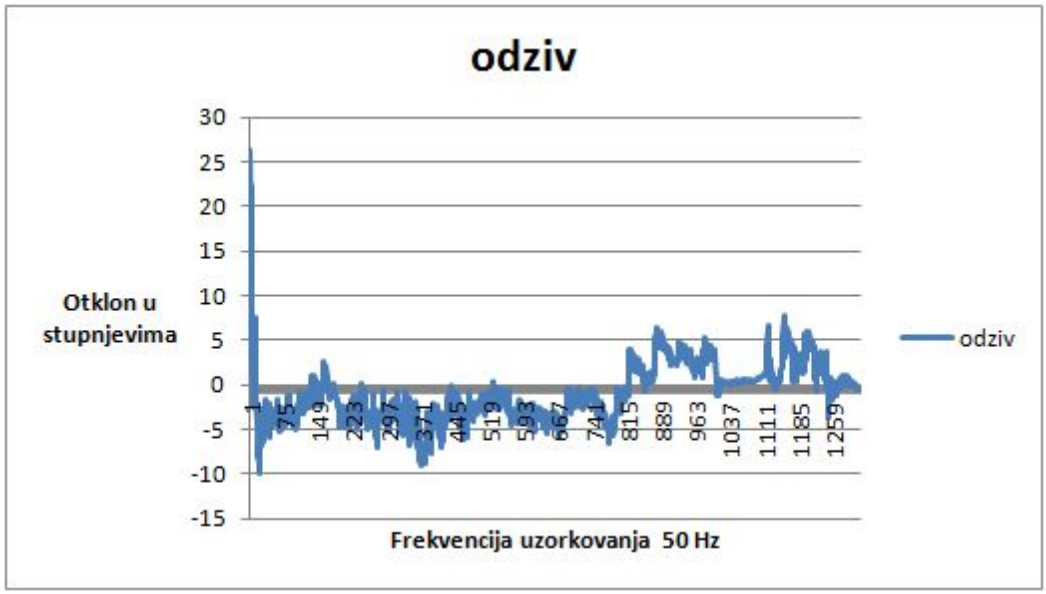


Slika 20. Odziv sustava nakon kretanja 8cm u lijevo s parametrima 3

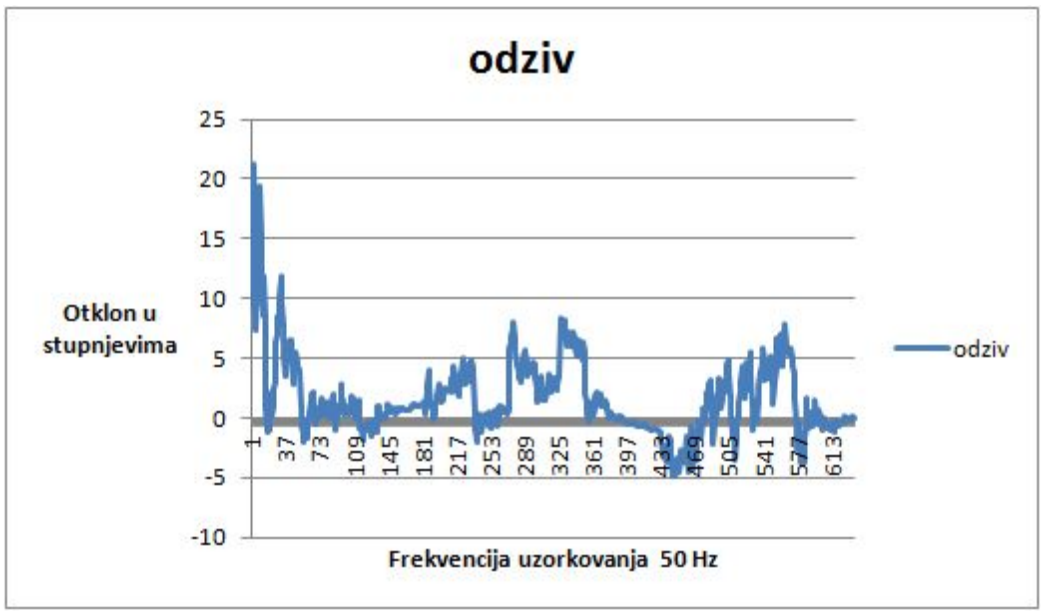


Slika 21. Odziv sustava nakon kretanja 15cm u lijevo s parametrima 3

Parametri regulatora 4: kP = 15, kI = 0.5, kD = 15, kPcons = 8, kDcons = 0.3, kIcons = 12

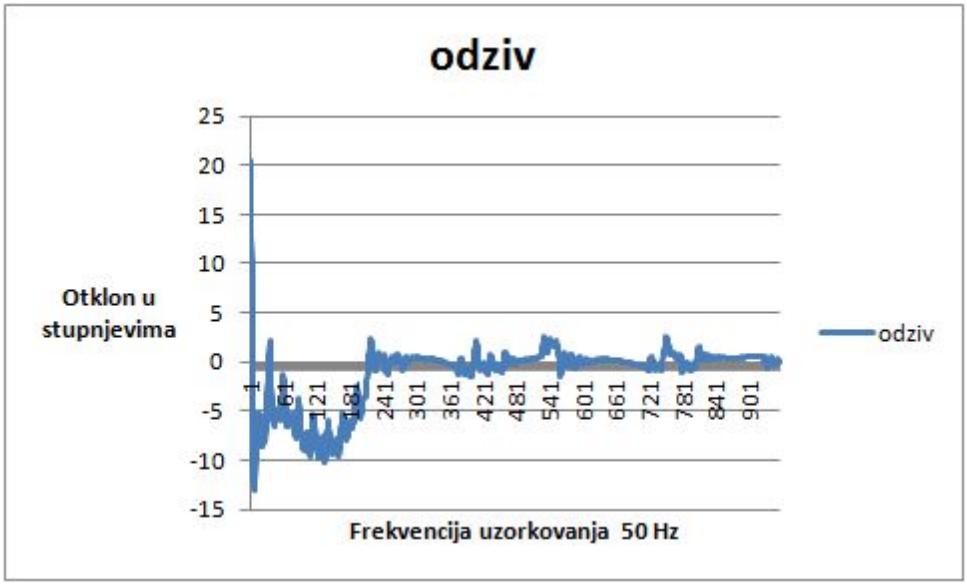


Slika 22. Odziv sutava nakon kretanja 17cm u lijevo s parametrima 4

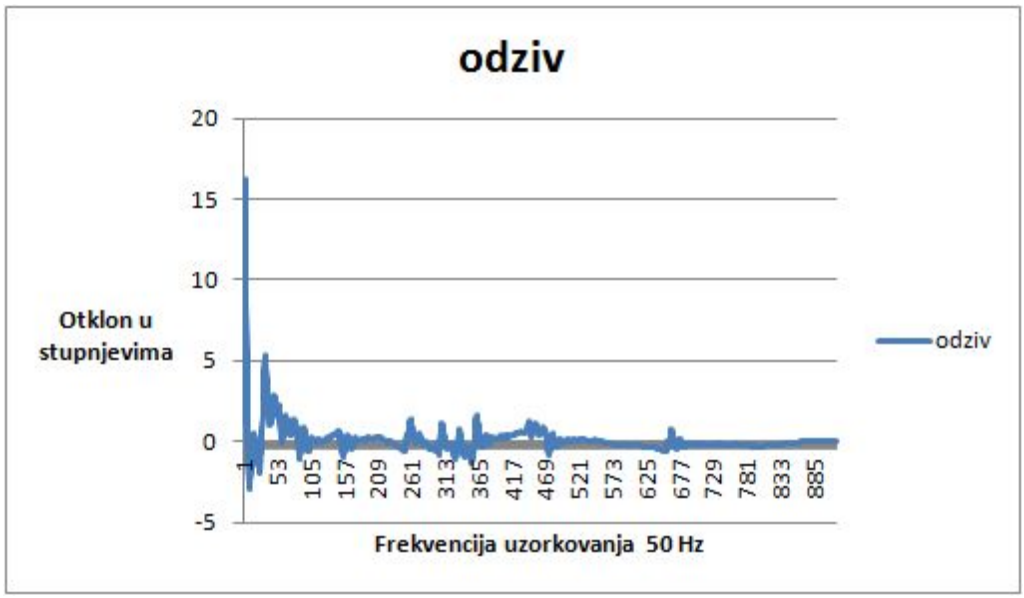


Slika 23. Odziv sustava nakon kretanja 2cm u desno s parametrima 4

Testiranje s drugom verzijom njihala, parametri regulatora 1:

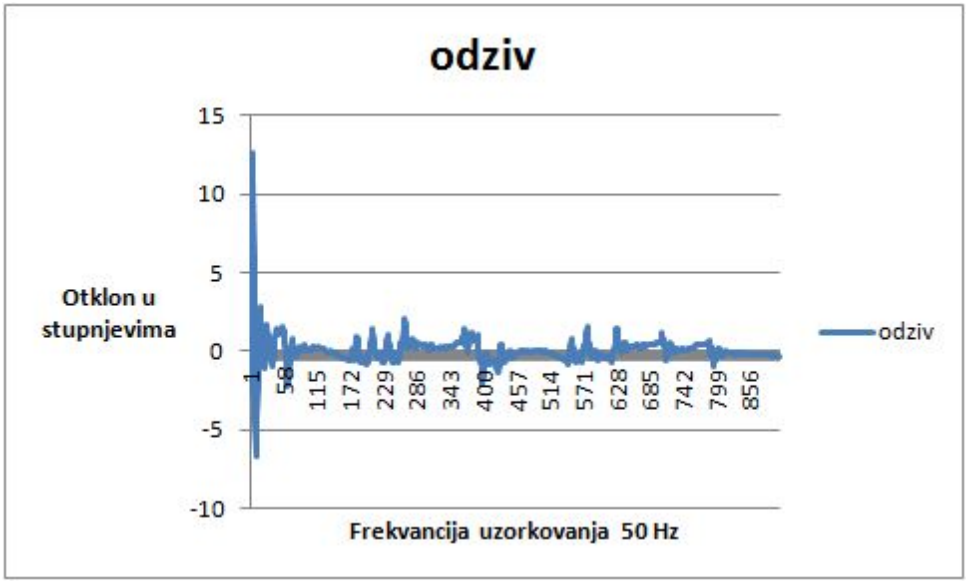


Slika 24. Odziv sustava s 2. verzijom njihala nakon pomaka 6cm lijevo, parametri regulatora 1

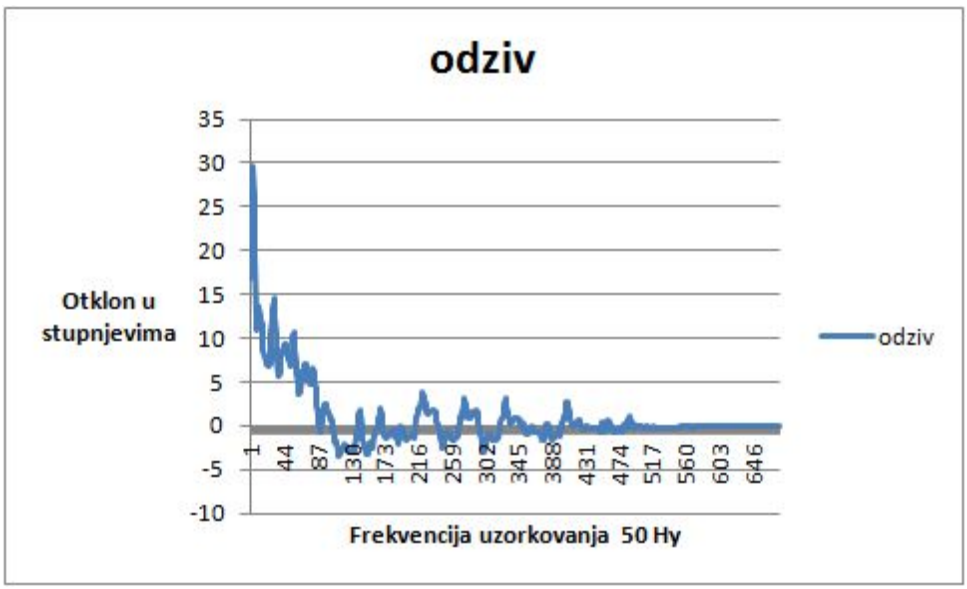


Slika 25. Odziv sustava s 2. verzijom njihala nakon pomaka 3cm desno, parametri regulatora 1

Parametri regulatora 4: kP = 30, kI = 1, kD = 30, kPcons = 25, kDcons = 1.1, kIcons = 28



Slika 26. Odziv sustava s 2. verzijom njihala nakon pomaka 10cm desno, parametri regulatora 4



Slika 27. Odziv sustava s 2. verzijom njihala nakon pomaka 2cm lijevo, parametri regulatora 4

5. ZAKLJUČAK

Projektni zadatak bio je zahtjevan po pitanju izrade sklopovlja a i programske podrške, spojiti sve korištene tehnologije u jedan sustav bilo je vrlo izazovno. Uz malo prethodnog znanja o Arduinu/Croduinu bilo je potrebno puno truda da bi se projekt završio. Važna stavka je da zadani sustav radi u stvarnom vremenu tako da su sklopovlje i program morali biti vrlo dobro prilagođeni za takav rad. Sustav ne radi savršeno jer bi to bilo iznimno teško napraviti. Mehaničke vibracije, odstupanje senzora, njihala i motora te povremeno prekidanje komunikacije senzora i Croduina pokušali smo smanjiti koliko je to bilo moguće. Uz sve to, zadovoljni smo radom sustava i smatramo da je projekt bio ključan u razvijanju novih znanja i upoznavanju novih tehnologija. Projekt bi se mogao nadograditi i poboljšati uz pristup kvalitetnijem sklopovlju i korištenjem 3D printera za izradu njihala i sličnih komponenti.

LITERATURA

[1] "Pendulum". Miriam Webster's Collegiate Encyclopedia. Miriam Webster. 2000. p. 1241. ISBN 0-87779-017-5. , prezeto sa https://en.wikipedia.org/wiki/Pendulum , dostupno 12. rujna 2016.
[2] http://ctms.engin.umich.edu/CTMS/Content/Activities/figures/pendulum.png, dostupno 12. rujna 2016.
[3] https://en.wikipedia.org/wiki/Foucault_pendulum , dostupno 12. rujna 2016.
[4] http://ieeexplore.ieee.org/document/876661/ , dostupno 12. rujna 2016.
[5] http://soil.co.uk/products/inclination/inverted-hanging-pendulum-systems/ , , dostupno 12. rujna 2016.
[6]http://klinikrobot.com/components/com_virtuemart/shop_image/product/resized/Minebea_Astrosyn_55c9f997119ee_250x250.jpg , dostupno 14.rujna 2016.
[7] https://www.pololu.com/product/1182 , dostupno 14.rujna 2016.
[8] https://a.pololu-files.com/picture/0J4577.600x480.jpg?289b47dcf4261eda4da0c8afce618dfd , dostupno 14.rujna 2016.
[9] http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/step3/Combining-the-Accelerometer-and-Gyro/ , dostupno 14.rujna 2016.
[10] http://www.hotmcu.com/gy521-mpu6050-3axis-acceleration-gyroscope-6dof-module-p-83.html , dostupno 14.rujna 2016.
[11] https://images-na.ssl-images-amazon.com/images/I/71pdoX7o9VL._SX425_.jpg , dostupno 14.rujna 2016.
[12] https://e-radionica.com/hr/croduino-basic2.html , dostupno 14.rujna 2016.
[13] http://www.netokracija.com/wp-content/uploads/2015/03/croduino.jpg , dostupno 14.rujna 2016
[14] http://www.newegg.com/Product/Product.aspx?Item=N82E16817165036 , dostupno 14.rujna 2016.
[15] http://thumbs3.ebaystatic.com/d/l225/m/mMRX403QcROvHxBlBaS6_SQ.jpg , dostupno 14.rujna 2016.
[16] https://theccontinuum.files.wordpress.com/2012/09/gy-80.png , dostupno 15.rujna 2016.
[17] https://www.pololu.com/picture/0J3360.600.png?d94ef1356fab28463db67ff0619afadf , dostupno 19.rujna 2016.

[18] PID Control: A brief introduction and guide, using Arduino. https://*www.maelabs.ucsd.edu/mae156alib/**control/PID-Control**-Ardunio.pdf* , *dostupno 20. kolovoza 2016.*