Automatsko podešavanje frekvencije pobude kapacitivnog senzora položaja

Mičić, Vladimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Automatsko podešavanje frekvencije pobude kapacitivnog senzora položaja

Završni rad

Vladimir Mičić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 28. rujna 2017.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Vladimir Mičić
Studij, smjer:	Preddiplomski sveučilišni studij Elektrotehnike
Mat. br. studenta, godina upisa:	3940, 2014.
OIB studenta:	51173785314
Mentor:	Doc.dr.sc. Davor vinko
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Automatsko podešavanje frekvencije pobude kapacitivnog senzora položaja
Znanstvena grana rada:	
Predložena ocjena završnog rada:	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: bod/boda Postignuti rezultati u odnosu na složenost zadatka: bod/boda Jasnoća pismenog izražavanja: bod/boda Razina samostalnosti: razina
Datum prijedloga ocjene mentora:	
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

Sveučilište Josipa Jurja Strossmayera u Osijeku

ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28. rujna 2017.

Ime i prezime studenta:	Vladimir Mičić
Studij:	Preddiplomski sveučilišni studij Elektrotehnike
Mat. br. studenta, godina upisa:	3940, 2014.
Ephorus podudaranje [%]:	

Ovom izjavom izjavljujem da je rad pod nazivom: **Automatsko podešavanje frekvencije pobude kapacitivnog senzora položaja**

izrađen pod vodstvom mentora Doc.dr.sc. Davora Vinka.

Moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj:

1. UVOD

Završni rad "Automatsko podešavanje frekvencije pobude kapacitivnog senzora položaja" omogućuje nam iskorištavanje prethodno napisanog koda u programu “Arduino IDE” kako bi se implementacijom spomenutog koda pomoću mikroupravljača Arduino Nano generirale određene frekvencije na kojima se pobuđuje kapacitivni senzor položaja te kako bi se mogla izmjeriti promjena napona na sklopu uzrokovana pobudom sklopa.

Osim kapacitivnih senzora položaja koji se primjenjuju i kod metalnih i kod nemetalnih materijala, za razne druge primjene postoje i drukčiji senzori, npr. induktivni senzori za metalne materijale, magnetni senzori koji se koriste kada induktivni blizinski senzori iscrpe svoje mogućnosti, opto-električni senzori koji imaju veći domet od dosad navedenih blizinskih, laserski senzori koji se koriste kad treba obuhvatiti male objekte ili je potrebna vrlo precizna detekcija objekta, akcelerometri, potenciometri, ultrazvučni, radarski i brojni drugi.

Kod odabranog kapacitivnog senzora položaja prvo je potrebno odabrati odgovarajuću frekvenciju rada, a tu frekvenciju generira se pomoću Arduino Nano razvojne pločice.

Senzor se spaja, s jedne strane na Arduino pločicu, a s druge na osciloskop pomoću kojeg ćemo očitavati grafove i spremati snimke zaslona raznih prikaza na osciloskopu. [1], [4]

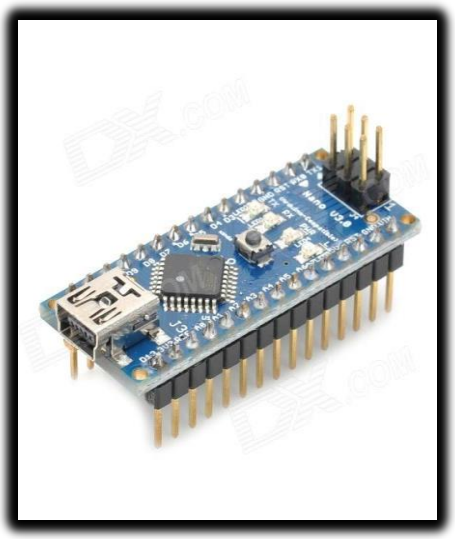
1.1. Zadatak rada

Zadatak završnog rada je razviti algoritam za automatsko podešavanje frekvencije pobude za kapacitivni senzor položaja. Razvijeni algoritam implementirati u odgovarajući sklop i testirati njegov rad kroz laboratorijska mjerenja.

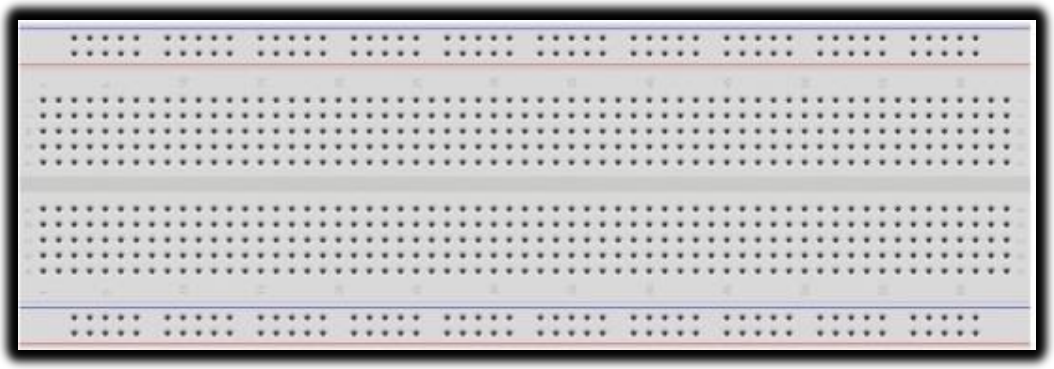
2. PPRICIP RADA

2.1. Izgled komponenata

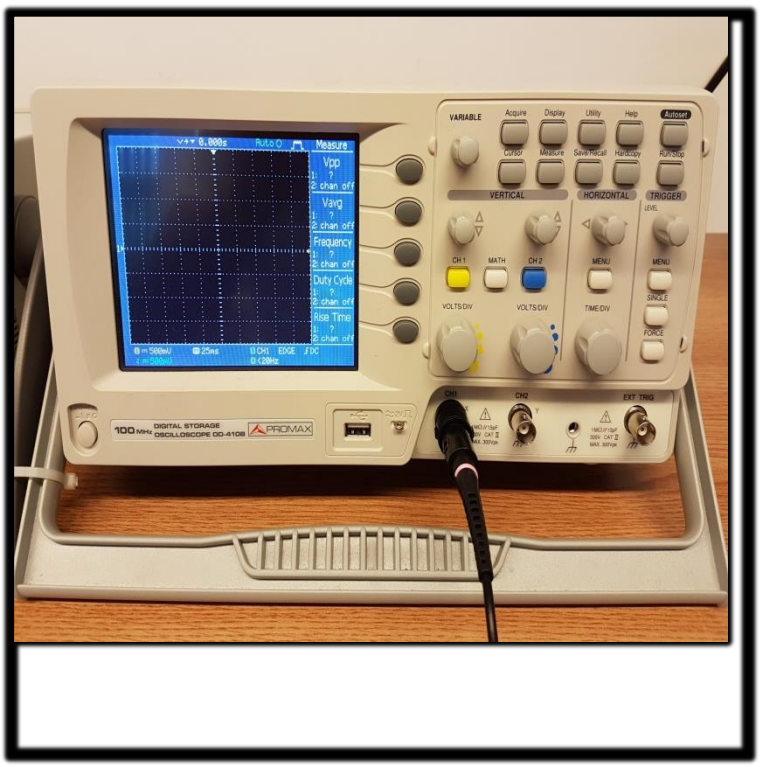
Za mikrokontroler korišten je Adruino Nano, točnije ATmega 328 na slici 2.1. Arduino je spojen na protoboard na slici 2.2. niže, a osciloskop na slici 2.3. niže je model korišten za mjerenja.



Slika 2.1. Arduino Nano

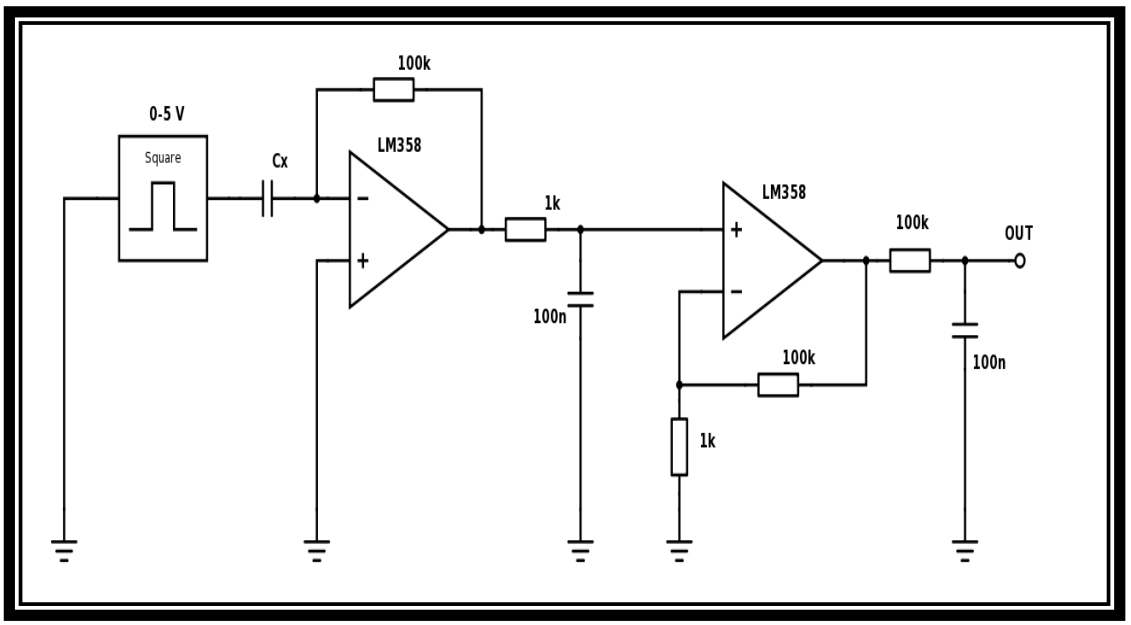


Slika 2.2. Protoboard



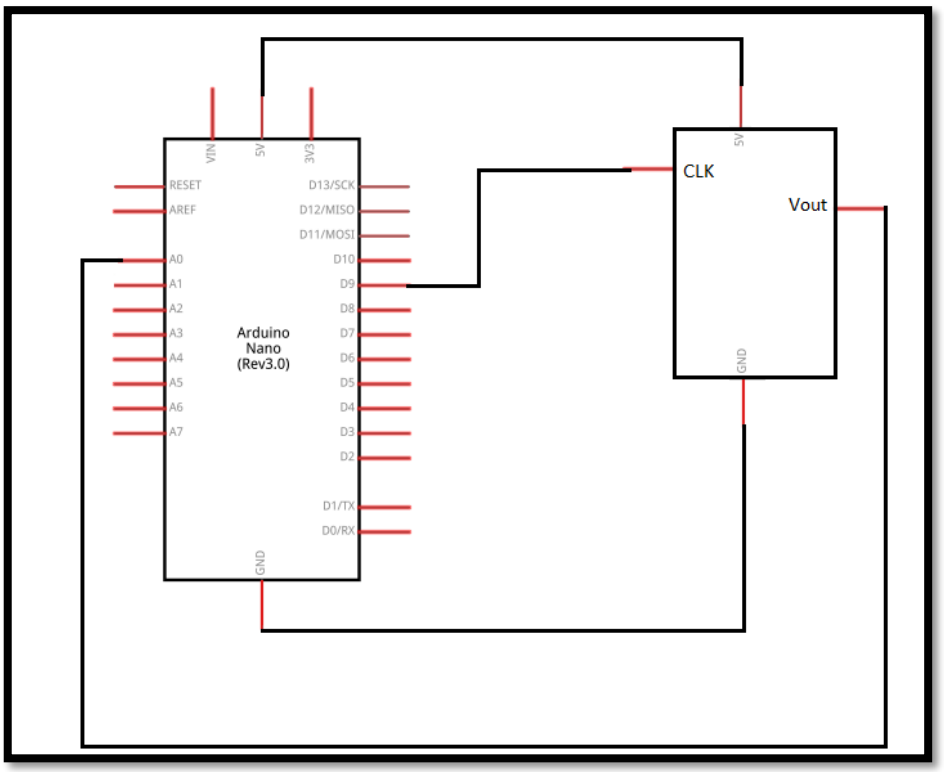
Slika 2.3. Osciloskop

2.2. Spajanje sklopa s ostalim dijelovima

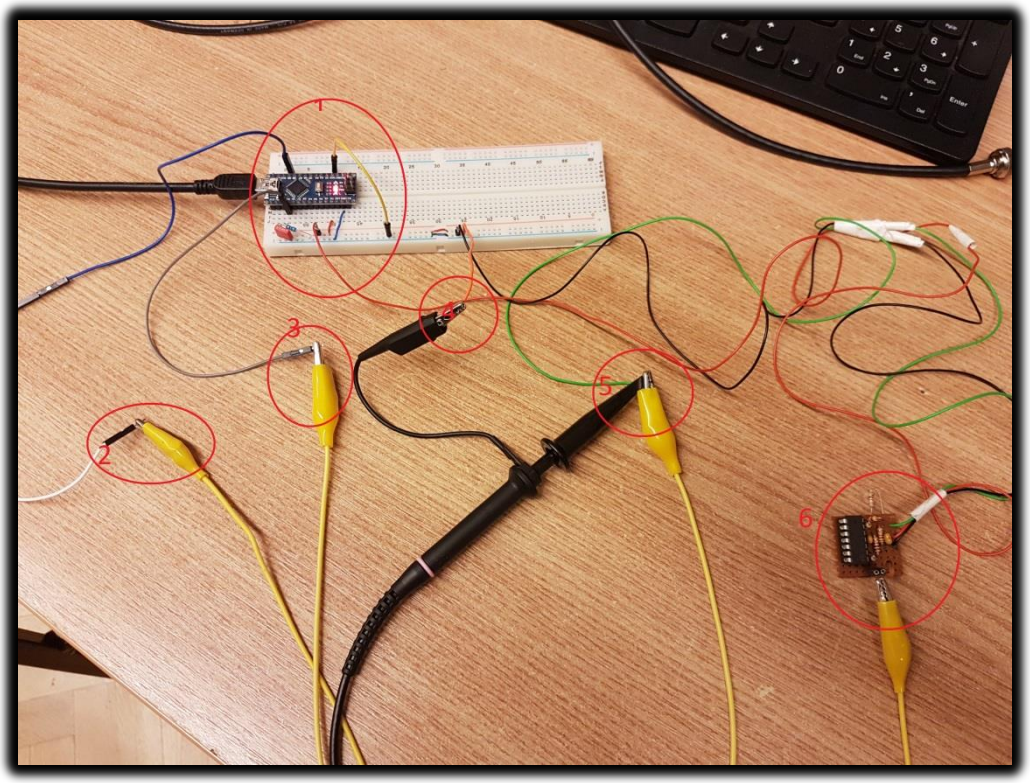


Slika 2.4. Shema sklopa na kojemu mjerimo napon.

Na slici 2.4. *Cx* je promijenjivi kondenzator, a ostatak sklopa služi za pretvorbu kapaciteta *Cx* u istosmjerni napon na izlazu. Kako se *Cx* mijenja, tako će se i mijenjati izlazni napon (*OUT*).



Slika 2.5. Shema spoja Arduino Nano pločice i senzora položaj [5]



Slika 2.6. Spoj Arduina, sklopa te potrebnih pomagala za mjerenje.

Povezanost komponenata je vidljiva na slikama 2.4. i 2.5.

Crvenom bojom uokvireni spojevi:

1. Arduino Nano napajan preko usb-a s računala +5 V; ground je s arduina spojen na masu protoboarda; na pinu D9 se očitavaju frekvencije koje su generirane pomoću arduina: na pinu A0 arduino mjeri napon sklopa te ga ispisuje na serial monitoru.
2. Pin D9 spojen krokodilkama na kondenzator sa sklopa.
3. Pin A0 krokodilkama spojen na žicu koja daje napon sklopa.
4. Sonda osciloskopa spojena na minus protoboarda.
5. Sondom osciloskopa prihvaćena žica koja daje napon sa sklopa.
6. Sklop na kojem mjerimo napon, crvena žica je napajanje sklopa s +5 V.

2.3. Opis komponenata

Arduino Nano

Arduino Nano je elektronska pločica s mikrokontrolerom na sebi (mali procesor), bazirana na ATmega 328. Arduino Nano napaja se pomoću USB – a, 6 – 20 V (neregulirano vanjsko napajanje) ili 5 V (regulirani vanjski izvor napajanja).
Arduino je nastao 2005. godine i potpuno je open source. Hardware arduina je strujni krug koji se može elektronički programirati, a software se naziva IDE i pomoću njega upravljamo pločicom.
Dvije temeljne petlje koje arduino program mora sadržavati su ‘setup’ i ‘loop’, gdje se ‘setup’ dio izvodi na početku rada ili kod resetiranja arduina, a ‘loop’ je beskonačna petlja u kojoj je glavni sadržaj programa.

Protoboard

Protoboard ili eksperimentalna pločica (slika 2.2.) je elektronički povezan sklop koji služi kao temeljno postolje za spajanje elektroničkih elemenata u strujni krug bez lemljenja.
Uobičajeni izgled pločice jest plastična vanjština s određenim brojem utora koji služe kao točke povezivanja; unutarnja strana pločice se povezana, a izvana se povezuje žicama (*eng. Jumper wires*).
S krajnje dvije strane nalaze se ‘+’ (najčešće crvena) i ‘-‘ (najčešće crna) linije.
[6]

2.4. Analiza rada

Sklop je baziran na Arduino Nano razvojnoj pločici. Arduino je razvojna platforma otvorenog koda, a temelji se na hardveru i softveru jednostavnim za korištenje, najčešće Atmel mikroupravljačima i Arduino IDE razvojnom okruženju. Programski jezik korišten u Arduino IDE je prilagođena verzija C++ programskog jezika.

Arduino Nano je mala razvojna pločica dimenzija 18x45 mm sa 22 digitalna ulazno-izlazna pina i 8 analognih. Bazira se na Atmel ATmega328 mikroupravljaču te ima 32 KB memorije (od kojih je 2 KB zauzeto bootloader-om), a ugrađen je i kvarcni kristal frekvencije 16 MHz. S računalom se povezuje putem USB mini-B kabla te radi na 5 V, a ima i mogućnost spajanja vanjskog izvora napona 7-12 V.

Sklop je složen i programiran da na digitalni pin 9 (D9) daje izlaznu frekvenciju raspona 10 kHz do 500 kHz, na analognom pinu 0 (A0) mjeri ulazni napon, a trenutnu izlaznu frekvenciju i izmjereni napon prikazuje na serial monitoru.

Postoji više načina za tvorbu pulsnog signala određene frekvencije na izlazu, ali postupak se svodi na mijenjanje stanja odabranog izlaza iz logične „0“ u logičku „1“ i obrnuto u određenom vremenu. Tako se, primjerice, pri frekvenciji od 1 Hz na izlazu u jednoj sekundi može primijetiti jedna logička „0“ i jedna logička „1“, pri frekvenciji od 2 Hz pojavljuju se dvije „0“ i dvije „1“ naizmjence itd. Za jednostavnije primjere za određivanje vremena trajanja „0“ i „1“ može se koristiti funkcije „delay()“ i „delayMicroseconds()“. Problem korištenja ovih funkcija je što one „pauziraju“ izvođenje ostatka programa pa ako se sklop koristi za još neku funkciju mora se uzeti u obzir i vrijeme potrebno za izvršavanje ostatka koda.

Drugi način je izravno korištenje mjeritelja vremena („timera“) i brojača („countera“) koji su ugrađeni u ATmega328 mikroupravljač. U navedeni mikroupravljač ugrađena su tri timera/countera, od kojih su dva 8-bitni, a jedan je 16-bitni (timer 1). Jedna od razlika između 8-bitnog i 16-bitnog timera/countera je u tome što 8-bitni može mjeriti ili brojati do 256, a 16-bitni do 65536 što omogućuje veću rezoluciju ili dulje brojanje. Svi timeri/counteri ovise o signalu takta sustava, koji je na većini Arduino pločica na frekvenciji od 16MHz, što znači 16.000.000 otkucaja u sekundi. Timeri/counteri se mogu postaviti u nekoliko načina rada pomoću TCCRx registra (x označava broj timera/countera koji se podešava). Za ovu primjenu odabran je CTC („clear timer on compare match“) način rada prilikom kojeg se vrijednost brojača poništava svaki put kada dosegne krajnju postavljenu razinu (vrijednost).

CTC način rada se postavlja tako da se u WGM12 bit TCCRx registra upiše logička „1“, a ostali WGM bitovi ostaju „0“. Krajnja vrijednost brojača postavlja se upisivanjem željene vrijednosti u OCR1A registar (maksimalno 256 ili 65536, ovisno koji timer/counter se koristi).

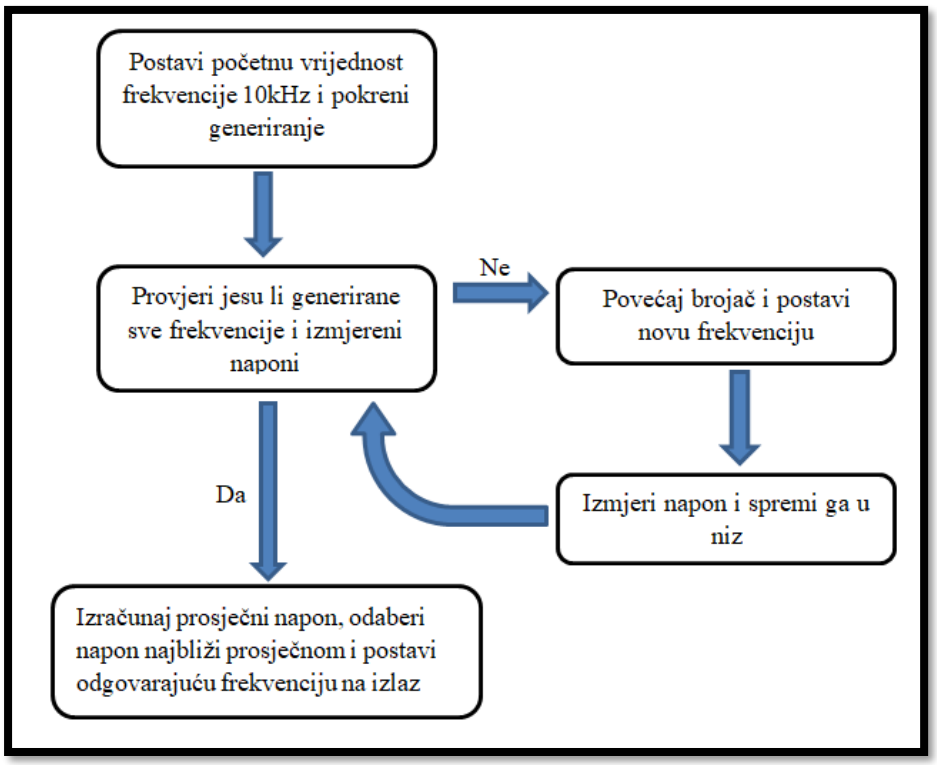
Kako je već navedeno, Arduino radi na frekvenciji od 16 MHz, što znači da se događa jedan otkucaj takta svakih 0,0625µs. U teoriji, to je najmanje moguće vrijeme za izvršavanje neke naredbe. Kako je za tvorbu pulsnog signala potrebna jedna „0“ i jedna „1“ potrebno je minimalno dva otkucaja takta, što znači da je maksimalna moguća frekvencija izlaza 8 MHz. Ako će brojač brojati svaki otkucaj takta najmanja frekvencija koju može ostvariti je otprilike 488 Hz, jer maksimalno može brojiti do 65536. Kako bi se i druge, niže, frekvencije mogle ostvarivati koristi se „prescaler“ ugrađen u svaki timer. Vrijednosti prescalera mogu biti 0, 1, 8, 64, 256, 1.024 ili vanjski izvor takta, a podešavaju se pomoću CS (clock select) bitova TCCR1B registra. Kako bi se dobila krajnja vrijednost timera (koja se upisuje u OCR1A registar) frekvencija sustava (16 MHz) se dijeli s vrijednošću prescalera (u ovom slučaju 1) te s dvostrukom vrijednošću željene krajnje frekvencije. Ako se želi generirati frekvencija od 10 kHz u OCR1A registar se tako upisuje vrijednost 800, a timer će u jednoj sekundi 20.000 puta odbrojiti do 800 (potrebno je 10.000 puta za tvorbu logičke „0“ i 10.000 puta za tvorbu logičke „1“).

Upisivanjem logičke „1“ u bit COM1A0 registra TCCR1A omogućava se promjena logičkog stanja odgovarajućeg izlaznog pina (OC1A, PortB pin 1 ili Ardino pin 9) pri svakom dostizanju krajnje postavljenje vrijednosti timera (vrijednosti upisane u registar OCR1A).

Mjerenje napona odrađuje se pomoću „analogRead“ funkcije, odnosno pomoću ugrađenog 10-bitnog analogno-digitalnog pretvarača (ADC). Očitana vrijednost tako iznosi između 0 i 1023. Tu vrijednost potrebno je pretvoriti u vrijednost napona tako da se pomnoži s 0,0048828125 (5/1024= 0,0048828125). Vrijednosti napona i frekvencije ispisuju se na serial monitoru funkcijom Serial.print(). [1], [2], [5]

2.5. Objašnjenje programskog koda

Kako bi se lakše razumio tok programa na slici 2.7. prikazan je dijagram toka:



Slika 2.7. Dijagram toka programa

Na početku programskog koda obično se navode korištene biblioteke. U ovom slučaju nisu korištene dodatne biblioteke pa su odmah definirane globalne varijable koje se koriste. Prva je niz koji sadržava vrijednosti potrebnih djelitelja za generiranje definiranih frekvencija.

```
// djelitelji za postavljanje frekvencije
long  divisor[] = {800, 400, 267, 200, 160, 133, 114, 100, 89, 80,
                   73, 67, 62, 57, 53, 50, 47, 44, 42, 40,
                   38, 36, 35, 33, 32, 31, 30, 29, 28, 27,
                   26, 25, 24, 23, 22, 21, 20, 19, 18, 17,
                   16
                  };
```

Slika 2.8. Definiranje niza „*divisor[]*“

Za odabir optimalne frekvencije planiran je korak od 10 kHz te testiranje frekvencija u rasponu od 10 kHz do 500 kHz, a optimalna frekvencija je ona uz koju je izmjereni napon u srednjoj vrijednosti. Obzirom da je generiranje izlazne frekvencije vezano uz dijeljenje

radnog takta mikroprocesora, u ovom slučaju 16 MHz, ne mogu biti generirane točno sve željene frekvencije. Tako primjerice frekvenciju od točno 30 kHz nije moguće ostvariti jer se frekvencija od 16000000 Hz ne može cjelobrojno podijeliti sa 30000 Hz. Dijeljenjem 16000000 sa 2x30000 dobije se broj 266,666, koji ako se zaokruži na najbliži sljedeći broj iznosi 267. Kad se za djelitelj odabere broj 267, a za prescaler 1 (kao u ovom slučaju) uz frekvenciju mikroupravljača od 16 MHz, izlazna frekvencija na pinu 9 iznosi 29962.55, što je nešto manje od planirane frekvencije od 30 kHz. Ako bi se, pak, za vrijednost djelitelja odabrao broj 266, uz iste uvjete bi izlazna frekvencija bila 30075,19 Hz. Obzirom da ne postoji manji prescaler od 1 nije moguće napraviti manji korak, odnosno promijeniti frekvenciju za manji iznos. Sve očekivane izlazne frekvencije upisane su u nizu *freq[]* prikazanom na slici 2.9.

```
// vrijednosti frekvencije
double  freq[] = {10000, 20000, 29962.55, 40000, 50000, 60150.38, 70175.44,
                  80000, 89887.64, 100000, 109589.04, 119402.99, 129032.26,
                  140350.88, 150943.40, 160000, 170212.77, 181818.18, 190476.19,
                  200000, 210526.32, 222222.22, 228571.43, 242424.24, 250000,
                  258064.52, 266666.67, 275862.07, 285714.29, 296296.30, 307692.31,
                  320000, 333333.33, 347826.09, 363636.36, 380952.38, 400000,
                  421052.63, 444444.44, 470588.24, 500000
                 };
```

Slika 2.9. Definiranje niza *freq[]*

Drugi problem nastaje kod generiranja frekvencija većih vrijednosti. Obzirom da su vrijednosti djelitelja sve manje i počinju se mijenjati, za samo jedan broj ne postoji način da se u manjem koraku promijeni frekvencija, odnosno ne postoji način za generiranje približnih iznosa željenih frekvencija. To je razlog zašto nizovi *djelitelj[]* i *freq[]* sadže samo 41 vrijednost, a ne očekivanih 50 (500000/10000=50).

Od postavljenih 41 vrijednost djelitelja samo ih 12 omogućava generiranje točnih frekvencija (10, 20, 40, 50, 80, 100, 160, 200, 250, 320, 400 i 500 kHz), a ostale imaju odstupanje od 30 do 3000 Hz, dok za 9 planiranih frekvencija, kako je već spomenuto, nije moguće generirati niti približnu frekvenciju (390 kHz, 430 kHz,…)

Osim spomenuta dva niza koriste se i druge globalne varijable koje služe za podešavanje vremenskog intervala promjene frekvencije, niza za spremanje očitanih napona i dr., a prikazane su na slici 2.10.

```
// varijable potrebne za manipulaciju podacima
unsigned long previousMillis = 0;
const long interval = 100;// interval promjene frekvencije
int i = 0;                // brojac redoslijeda frekvencija
const int sensorPin = A0; // pin za mjerenje napona
int sensorValue = 0;      // vrijednost izmjerena na pinu A0 (napona) 0-1023
float voltage = 0;        // vrijednost napona u decimalnom obliku 0-5V
int napon[41];         // niz izmjerenih napona
float absolute[41];    // niz razlika između prosječnog napona i svih napona
int najveci=-1;    // niz razlika između prosječnog napona i svih napona
int najmanji=1024;    // najmanja razlika između prosječnog napona i svih napona
int minJ = -1;       // mjesto u nizu u kojem je najmanji napon
float minDelta = 5;    // najmanja razlika između prosječnog napona i svih napona
int gotovo = 0;        // ako su sve frekvencije testirane gotovo = 1
```

Slika 2.10. Globalne varijable programskog koda

U prvom od dva obvezna dijela svakog Arduino koda, „*void setup()*“ funkciji, onemogućeni su svi prekidi naredbom „*cli()*“ kako bi se program neometano i neprekidno izvodio, započeta je serijska komunikacija za ispis rezultata u serial monitor naredbom „*Serial.begin(9600)*“ te je postavljen timer/counter 1 za generiranje izlazne frekvencije na izlaznom pinu (digitalni pin 9) prema slici 2.11.

```
void setup() {
  // početak serijske komunikacije
  Serial.begin(9600);
  //Onemogući prekide
  cli();
  // Izbriši vrijednosti iz registara Timera1
  TCCR1A = 0;
  TCCR1B = 0;
  // Postavi OCR1A (krajnju vrijednost): 16MHz/10KHz/2 = 800 koraka
  OCR1A = 800;
  // COM1A0 za promjenu logičkog stanja pina OC1A (pin9)
  // kad se dosegne krajnja vrijednost
  TCCR1A = _BV(COM1A0);
  // WGM12 za CTC način rada
  // CS10 za postavljanje prescalera 1
  TCCR1B = _BV(WGM12) | _BV(CS10);
  // Postavljanje OC1A (Pin 9/Port B Pin 1) kao izlaz
  DDRB |= _BV(1);
}
```

Slika 2.11. Funkcija „*void setup()*“

Naredbama *TCCR1A=0* i *TCCR1B=0* vrijednosti tih registara postavljene su u 0, odnosno izbrisane su sve (eventualne) ranije upisane vrijednosti. To su registri kojima se postavlja način rada timera/countera. U registar *OCR1A* upisuje se krajnja vrijednost brojača, tj. djelitelja ulazne frekvencije. Ako se upiše broj 800 generira se frekvencija od 10 kHz jer je frekvencija mikroupravljača 16 MHz, a prescaler 1. Kada se 16.000.000 podijeli s 1 i s 800 dobije se broj 20.000, a taj broj kazuje koliko se puta promijeni stanje izlaznog pina. Obzirom da je u jednom periodu frekvencije jedna logička „0“ i jedna logička „1“ dobiveni broj 20.000 se još dijeli s 2 te se dobije krajnja vrijednost izlazne frekvencije 10 kHz. Isti princip izračuna vrijedi i za ostale vrijednosti djelitelja (registra *OCR1A*). Upisivanjem logičke „1“ u *COM1A0* bit registra *TCCR1A* omogućava se promjena stanja izlaznog pina (pina *OC1A*, tj. pina 1 porta B ili Arduino digitalnog pina 9) kad brojač dosegne krajnju vrijednost postavljenu u *OCR1A* registru. Upisivanjem logičke „1“ *WGM12* bit registra postavlja se CTC način rada u kojem je krajnja vrijednost postavljena u registru *OCR1A*, a upisivanjem logičke „1“ u bit *CS10* istog registra postavlja se vrijednost prescaler-a na 1. Ako je prescaler postavljen na 1 trenutna vrijednost brojača povećava se za 1 svakim otkucajem takta miroprocesora, ako je postavljen na 8, vrijednost brojača se povećava svakim osmim otkucacajem takta mikroupravljača itd. Kako bi se omogućile promjene logičkog stanja pina, on mora biti definiran kao izlaz (izlazni pin), a to je napravljeno postavljanjem logičke „1“ na u bit 1 *DDR* („data direction register“) regista za port B (*DDRB*).

U drugom obveznom dijelu Arduino programa, funkciji „*void loop()*“ koja se opetovano izvodi, a prikazanoj na slici 2.12, napravljen je tok progama. [2], [5]

```
void loop() {                 // dio koda koji se stalno izvodi
  if (gotovo != 1) {          // ako još nije odabrana krajnja frekvencija
    unsigned long currentMillis = millis(); // trenutno vrijeme
    // ako je razlika trenutnog i prošlog vremena veća od postavljenog intervala
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
      moveUp(); // pokretanje funkcije za promjenu frekvencije
    }
    // očitavanje vrijednosti na analognom pinu 0,
    // očitavanje napona; vrijednosti od 0 do 1023;
    // 10-bitni ADC (analogno digitalni konverter)
    sensorValue = analogRead(sensorPin);
    voltage = sensorValue * 0.0048828125; // pretvori očitanu vrijednost u napon 0-5V
    napon[i] = sensorValue; // spremanje očitanog napona u niz
    // ispis u Serial monitor
    Serial.print("Napon: ");
    Serial.println(voltage);
    Serial.print("Frekvencija: ");
    Serial.println(freq[i]);
  }
}
```

Slika 2.12. Funkcija „*void loop()*“

Kada se pokrene, funkcija provjerava je li varijabla „*gotovo*“ postavljena u vrijednost „1“, odnosno je li program završio sa traženjem optimalne frekvencije. Ukoliko nije nastavlja se sa izvođenjem, odnosno provjerom je li prošlo 100 ms (ili više) od zadnjeg pokretanja petlje. Trenutno vrijeme se funkcijom „*millis()*“ upisuje u varijablu *currentMillis*. Varijabla *previousMillis* je u početku postavljena na 0 te se oduzima od *currentMillis*, i ako je razlika veća od 100ms trenutna vrijednost vremena upisuje se u varijablu *previousMillis* i poziva se funkcija „*moveUp()*“. Zatim se u varijablu *sensorValue* učitava vrijednost s pina A0, odnosno mjeri se ulazni napon. Očitana vrijednost bit će u rasponu 0 do 1023 kolika je rezolucija 10-bitnog analogno-digitalnog konvertera koji analogni napon pretvara u digitalnu vrijednost. Kako bi se znao točan napon očitana vrijednost se množi sa 0.0048828125 kako bi se dobila vrijednost od 0 do 5 (volta). Očitana vrijednost napona također se upisuje na svoje mjesto u nizu *napon[]* naredbom „*napon[i] = sensorValue*“. *Serial.print* funkcijama se zatim izračunata vrijednost napona i odgovarajuće (trenutna) vrijednost frekvencije ispisuju u serial monitor.

Na slici 2.13. prikazana je funkcija „*moveUp()*“ koja se, kako je navedeno, poziva ukoliko još nije pronađena optimalna izlazna frekvencija. [3], [5]

```
void moveUp() {              //funkcija za povećanje frekvencije
  i++;                       // povecaj brojač
  if (i > 41)
  {
    stopCounter();         // postoji 41 zadana frekvencija - zaustavi brojač
  }
  else {
    OCR1A = divisor[i];        // postavi novu krajnju vrijednost brojaca (timera)
  }
}
```

Slika 2.13. Funkcija „*moveUp()*“

U funkciji se vrijednost brojača koji broji koliko je puta funkcija pozvana povećava naredbom *i++*. Ako je ta vrijednost veća od 41 poziva se funkcija „*stopCounter()*“ prikazana na slikama 2.14. i 2.15. Ukoliko vrijednost brojača nije veća od 41 u *OCR1A* registar učitava se nova vrijednost, odnosno odgovarajuća vrijednost iz niza *divisor[]*, a time se postavlja nova krajnja vrijednost brojača timera za generiranje frekvencije. [5]

```
void stopCounter() {
  TCCR0B &= 0B00101000;   // zaustavi brojač - bitovi CS10, CS11 i CS12 su 0
  int naponi = 0;
  for (int k = 0; k < 41; k++)
  {
   if (napon[k]>najveci){
    najveci=napon[k];      // određivanje najvećeg napona
    }
    else if (napon[k]<najmanji){
    najmanji=napon[k];     // određivanje najmanjeg napona
    }
   }
  // izračun prosječne vrijednosti napona i pretvaranje u vrijednost između 0V i 5V
  double prosjek = (najveci+najmanji)/2;
  double prosjekIspis = prosjek * 0.0048828125;
  // ispis prosječnog napona u Serial monitor
  Serial.print("Prosjecni napon: ");
  Serial.print(prosjekIspis);
  Serial.println("V");
```

Slika 2.14. Prvi dio funkcije *stopCounter()*

U prvom dijelu ove funkcije timer 1 se zaustavlja upisivanjem logičkih „0“ u bitove *CS10*, *CS11* i *CS12* registra *TCCR0B* naredbom *TCCR0B &= 0B00101000*. Zatim se među svim izmjerenim naponima i spremljenim u niz *napon[]* traže najveći i namjanji unutar „*for*“ petlje. Dobiveni najmanji i najveći napon se sada zbrajaju i dijele s 2 kako bi se dobio prosječni te množe sa 0.0048828125 kako bi se dobila vrijednost napona od 0 V do 5 V. Prosječni izračunati napon se zatim ispisuje u serial monitor.

U drugom dijelu ove funkcije, prikazanom na slici 2.15., odabire se napon najbliži prosječnom izračunatom i na izlaz postavlja njemu odgovarajuća frekvencija iz niza *freq[]*. [5]

```
// izračun razlika između prosječnog napona i izmjerenih napona
for (int j = 0; j < 41; j++)
{
  float razlika = prosjek - napon[j];
  absolute[j] = abs(razlika);
  // pronalazak najmanje razlike napona
  if (minDelta > absolute[j]) {
    minDelta = absolute[j];
    // mjesto u nizu na kojem je najmanja razlika napona
    minJ = j;
  }
}
// završen odabir frekvencije
gotovo = 1;
// podesi odabranu frekvenciju
OCR1A = divisor[minJ];
// ispis u Serial monitor
Serial.print("Odabrana frekvencija: ");
Serial.print(freq[minJ]);
Serial.println("Hz");
TCCR1B =_BV(WGM12)|_BV(CS10); // ponovno pokretanje Timeral
```

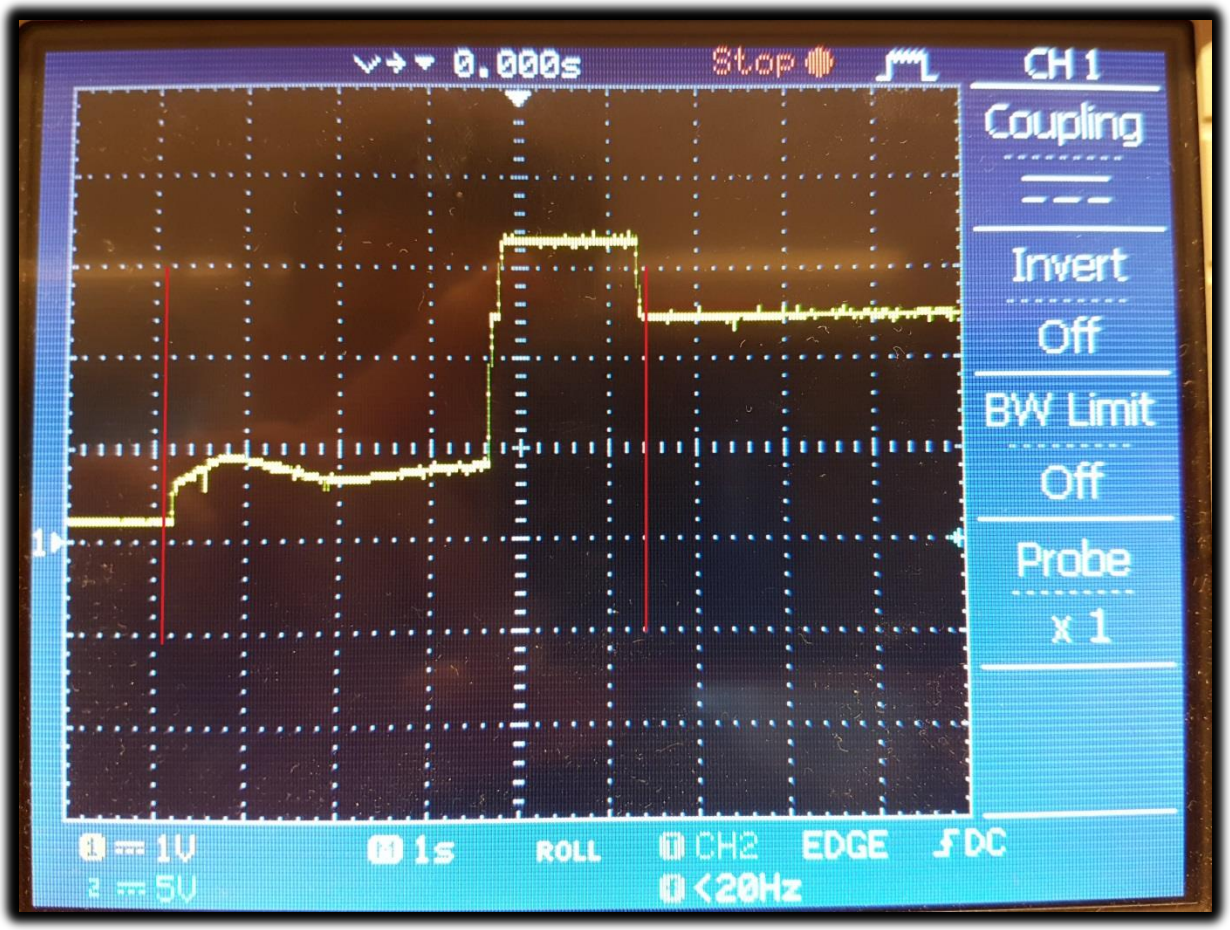
Slika 2.15. Drugi dio funkcije *stopCounter()*

U „*for*“ petlji se računa razlika između prosječnog napona i svakog izmjerenog napona. U niz *absolute[]* se upisuju apsolutne vrijednosti razlika te se svaka uspoređuje s varijablom *minDelta*. Ukoliko je apsolutna vrijednost razlike manja od varijable *minDelta*, u varijablu *minDelta* se upisuje nova manja vrijednost, a u varijablu *minJ* upisuje se vrijednost brojača kojim se prolazi kroz niz, odnosno bilježi se mjesto u nizu na kojem je najmanja razlika prosječnog i izmjerenog napona. U varijablu „*gotovo*“ tad se upisuje vrijednost 1 kako bi se

označilo da je pronađen optimalni (srednji) napon. Djelitelj potreban za generiranje optimalne frekvencije se unutar niza „*divisor[]*“ nalazi na istom mjestu kao i optimalni napon unutar niza „*napon[]*“ i vrijednost frekvencije unutar niza „*freq[]*“, dakle na mjestu koje je upisano u varijablu „*minJ*“. U registar *OCR1A* se stoga upisuje vrijednost djelitelja „*divisor[minJ]*“. Odabrana (optimalna) frekvencija se zatim ispisuje u serial monitor, a timer/counter 1 se ponovno pokreće upisivanjem logičkih „1“ u bitove „*WGM12*“ i „*CS10*“ registra *TCCR1B* kako bi se na pin 9 generirao signal odabrane frekvencije.

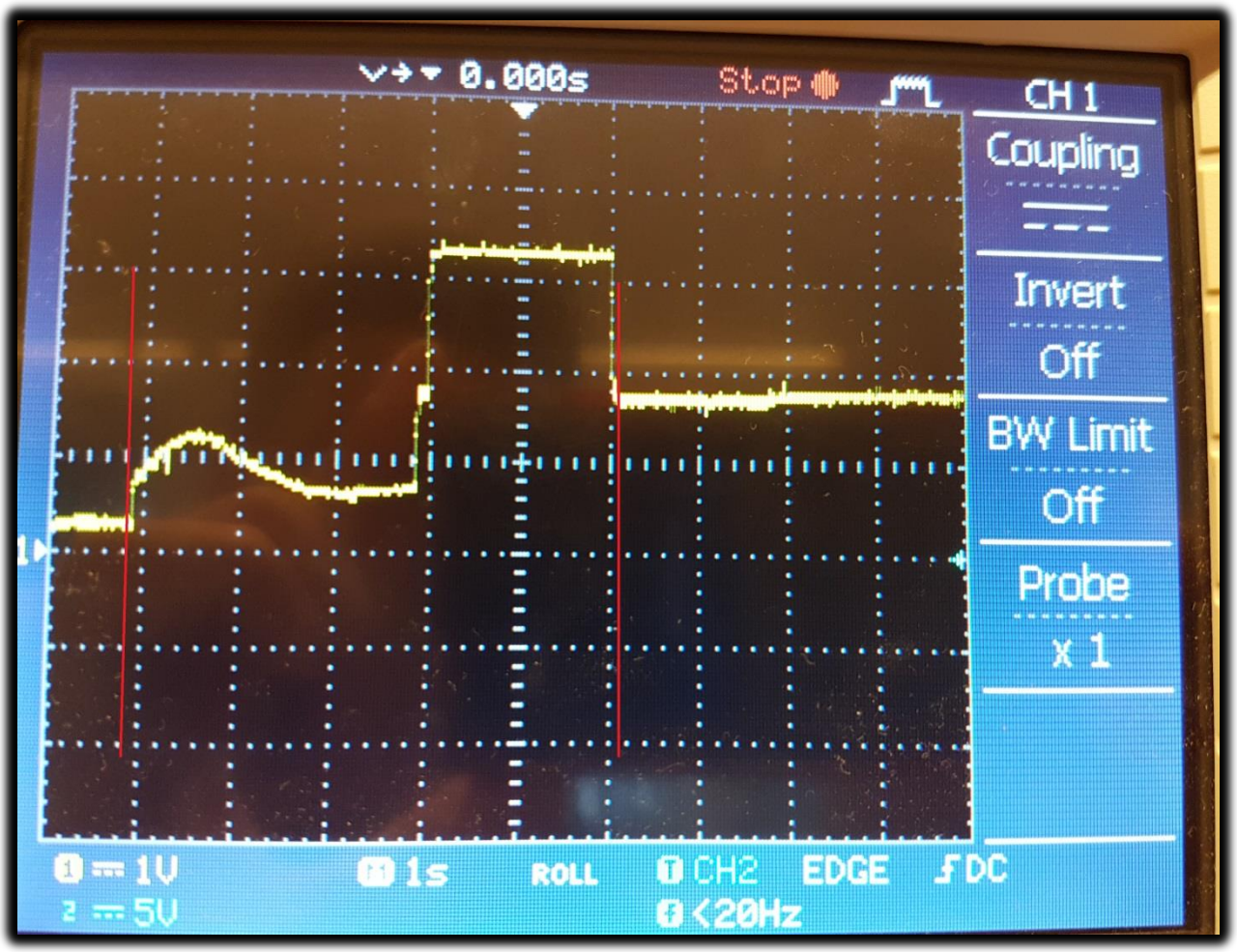
3. MJERENJA U LABORATORIJU

Mjerenja u laboratoriju su izvršena osciloskopom „PROMAX“ sa slike 2.3. uz promjenu kapaciteta (korišteno je 6 različitih kondenzatora), stoga se niže vidi slika grafa prikaza napona za svaki kondenzator. Slike prikazuje najvišu i najnižu vrijednost izlaznog napona senzora položaja pri promjenama frekvencija od 10 kHz do 500 kHz uz korištenje pojedinog kondenzatora te srednji napon koji se javlja pri „najpogodnijoj“ frekvenciji.



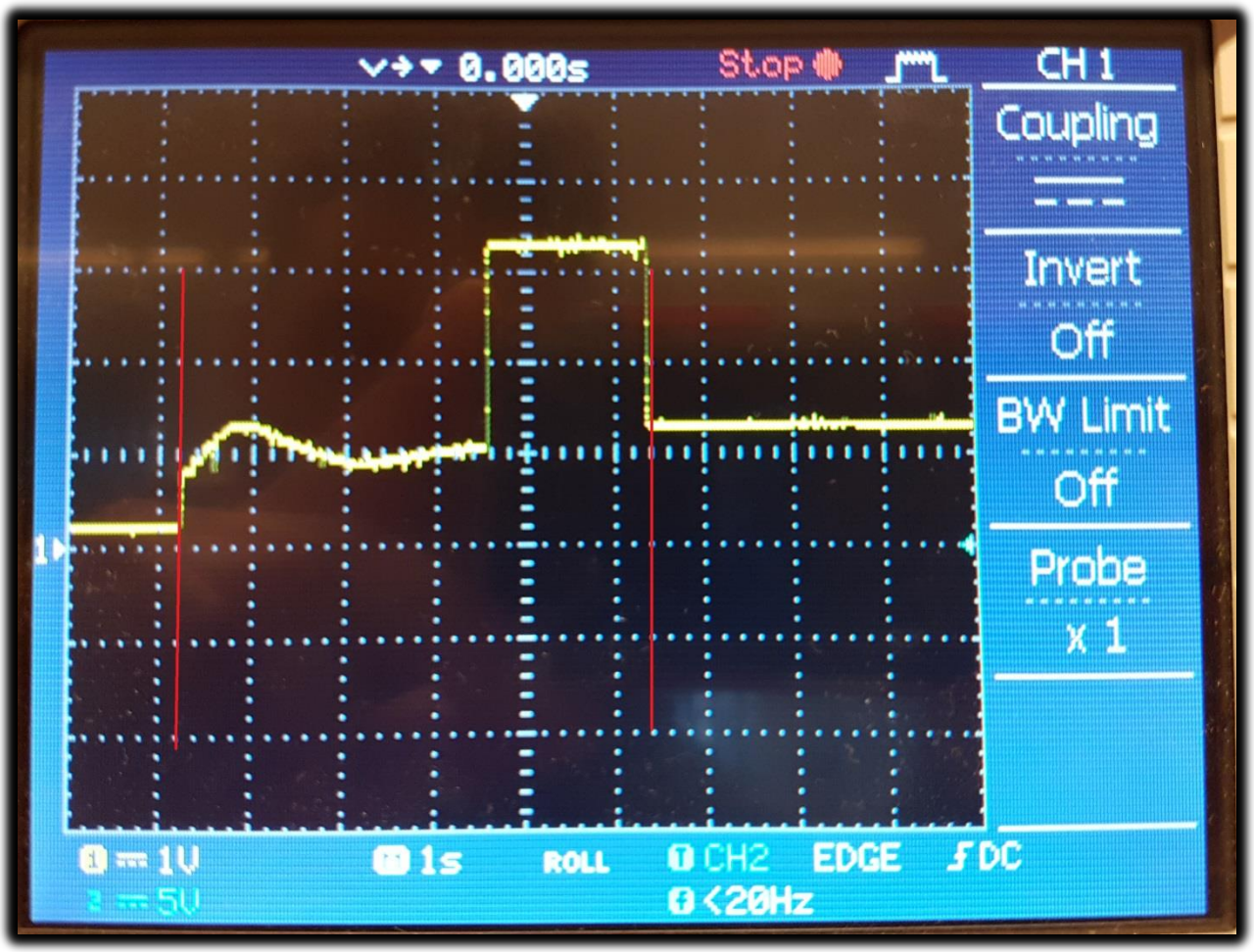
Slika 3.1. Izlazni napon senzora položaja; koristeći kapacitet 3,3 pF

Slika 3.1. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 3,3 pF. Gledano slijeva prva crvena okomita linija označava početak testiranja frekvencija, tj. signal sa sklopa, napon mijenja vrijednost do druge crvene okomite linije koja predstavlja početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji).



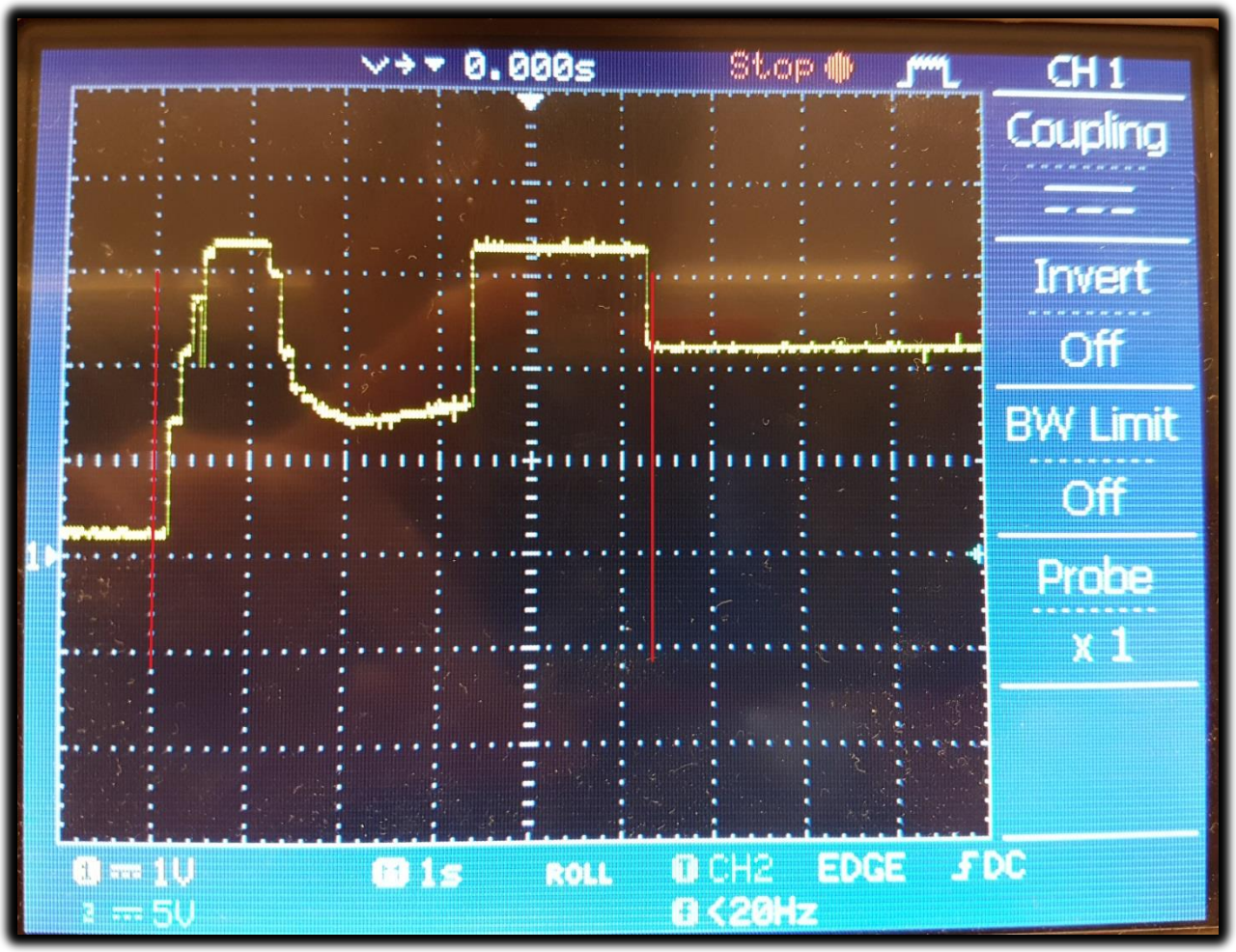
Slika 3.2. Izlazni napon senzora položaja; koristeći kapacitet 5 pF

Slika 3.2. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 5 pF. Gledano slijeva prva crvena okomita linija označava početak signala sa sklopa, nadalje napon blago raste, zatim opada te ponovno raste i doseže vršnu vrijednost. Druga crvena okomita linija predstavlja početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji).



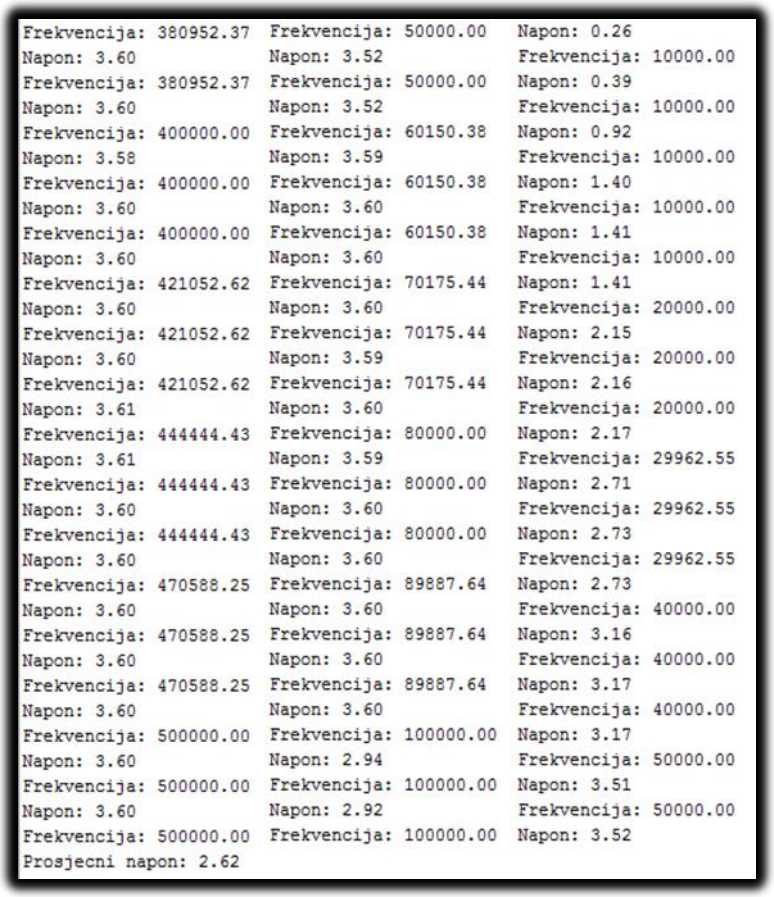
Slika 3.3. Izlazni napon senzora položaja; koristeći kapacitet 5,6 pF

Slika 3.3. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 5,6 pF. Gledano slijeva prva crvena okomita linija označava početak signala sa sklopa, napon blago raste, kratko opadne te ponovno raste do vršne vrijednosti. Druga crvena okomita linija predstavlja početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji).



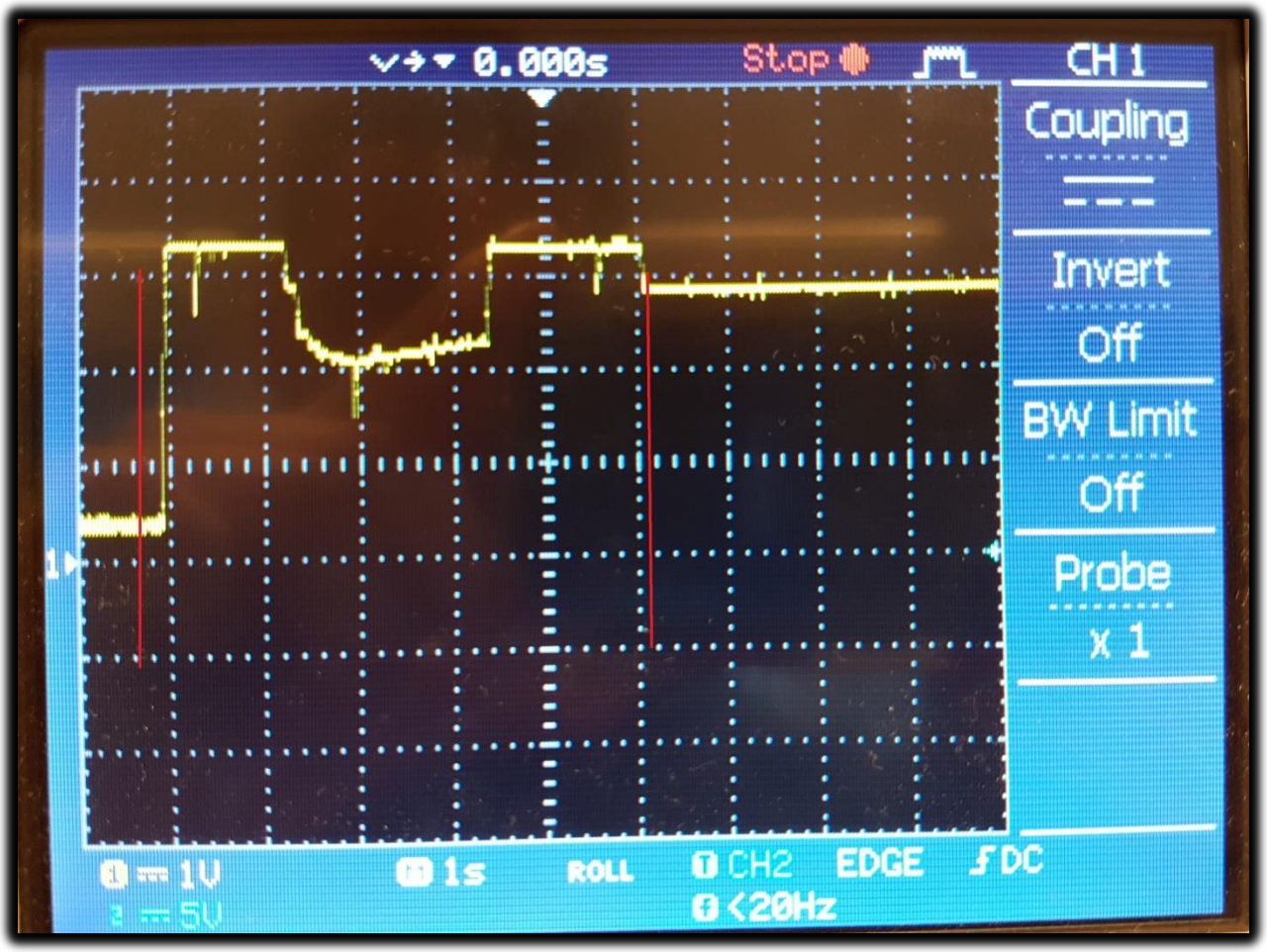
Slika 3.4. Izlazni napon senzora položaja; koristeći kapacitet 10 pF

Slika 3.4. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 10 pF. Gledano slijeva prva crvena okomita linija označava početak signala sa sklopa, napon raste do vršne vrijednosti, zatim opadne te ponovno raste do vršne vrijednosti. Druga crvena okomita linija predstavlja početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji). Najsličniji prikaz, onom željenom (idealnom), se dobije koristeći kapacitet 10 pF.



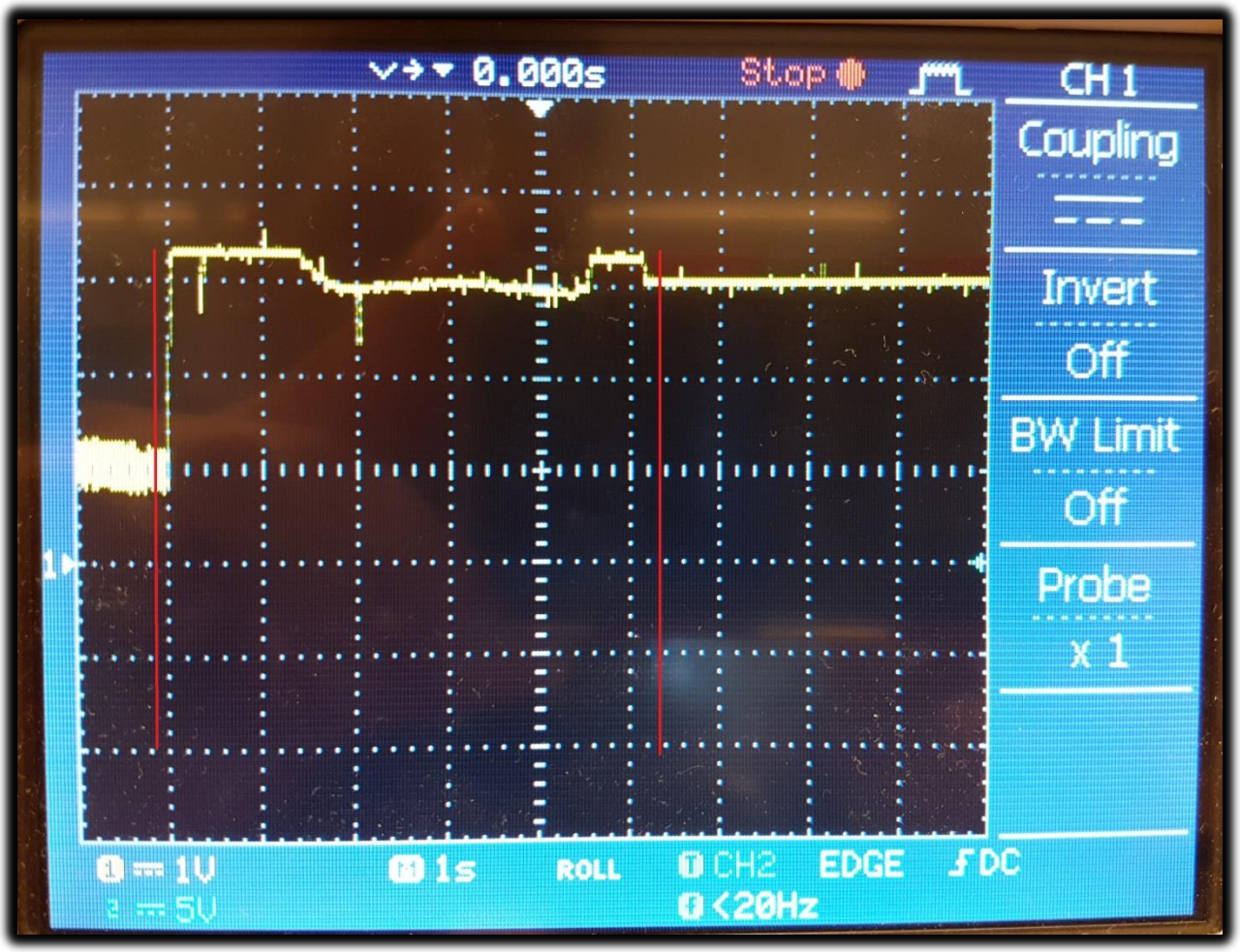
Slika 3.4.1. Mjerenje napona na zadanoj frekvenciji za kondenzator kapaciteta 10 pF

Slika 3.4.1. prikazuje primjer pridruživanja napona zadanoj frekvenciji (na slici se nalazi nekoliko reprezentativnih mjerenja, nisu sva sadržana zbog opsega podataka) te određivanje prosječnog napona za kondenzator kapaciteta 10 pF.



Slika 3.5. Izlazni napon senzora položaja; koristeći kapacitet 20 pF

Slika 3.5. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 20 pF. Gledano slijeva prva crvena okomita linija označava početak signala sa sklopa, napon raste do vršne vrijednosti, zatim opadne te ponovno raste do vršne vrijednosti. Druga crvena okomita linija predstavlja početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji).



Slika 3.6. Izlazni napon senzora položaja; koristeći kapacitet 75 pF

Slika 3.6. prikazuje izlazni napon senzora položaja kad se koristi kondenzator kapaciteta 75 pF. Gledano slijeva prva crvena okomita linija označava početak signala sa sklopa, napon slabo varira u vrijednosti i ne prikazuje se željeni valni oblik napona senzora položaja. Druga crvena okomita linija trebala bi predstavljati početak napona pri kojemu sklop optimalno radi, tj. srednji napon (najpogodniji), no pri korištenju kapaciteta 75 pF, sklop prestaje raditi optimalno (velik kapacitet).

4. ZAKLJUČAK

Na arduino je spojen sklop kojemu mjerimo napon na izlazu i pridružujemo ga pripadajućoj frekvenciji na kojoj je generiran te nakon obavljanja svih mjerenja napona na odgovarajućim frekvencijama, izabiremo najadekvatniju frekvenciju.

Frekvencija koju smo nazvali "najadekvatnija" je ona na kojoj dolazi do najveće promjene napona prilikom mijenjanja frekvencije, a na dobivenom grafu koji prikazuje ovisnost napona o frekvenciji nalazi se na dva mjesta, odnosno na padajućem i rastućem bridu karakteristike.

Mjerenja u laboratoriju rađena su pomoću osciloskopa, a sam sklop radi kao kapacitivni senzor položaja koji biva pobuđen generiranom frekvencijom.

Kod u programu Arduino IDE je pojednostavljena, tj. prilagođena verzija C++ za arduino; korišteni su timeri: odabran je CTC („clear timer on compare match") način kod kojeg se vrijednost brojača poništava uvijek kada dostigne krajnju postavljenu vrijednost.

S jedne strane, ovaj program ima nekoliko prednosti, a to su: automatsko generiranje frekvencija u zadanom opsegu, mjerenje napona te pridruživanje izmjerenih napona odgovarajućim frekvencijama, pronalazak „idealne" frekvencije na kojoj se sa sklopa očitava srednji napon i ispisi svih frekvencija te prosječnog napona. Ovaj program opravdava naziv rada jer pronalazi pobudu za idealan naponski odziv nekog sklopa. S druge strane, nedostatak ovog programa je njegova „grubost" u generiranju frekvencija. Ne mogu se generirati frekvencije dobivene računskim putem (teoretski), uvijek u praksi postoje gubitci i nekakva odstupanja. Pod terminom „grubost" misli se na korak između generiranja frekvencija koji ne može uvijek biti jednak zbog dijeljenja osnovne frekvencije na kojoj Atmega328 radi.

Povećanjem kapaciteta smanjuje se razlika između maksimuma i minimuma izlaznog napona senzora položaja, signal postaje „tuplji", što se vidi na slici 3,6, tj. pri kapacitetu od 75 pF, sklop prestaje raditi optimalno.

Program najbolje radi kada se koristi kondenzator kapaciteta 10 pF, odnosno, za tu vrijednost kapaciteta prikaz izlaznog napona senzora položaja je najsličniji željenom.

LITERATURA

[1] Uroš Peruško - "Digitalna elektronika" – prošireno izdanje, ŠKOLSKA KNJIGA, Zagreb

[2] https://e-radionica.com/en/ (pristup ostvaren 23. lipnja 2017.)

[3] https://www.arduino.cc/ (pristup ostvaren 8. i 23. Lipnja 2017.)

[4] http://gen.lib.rus.ec/search.php?req=arduino+nano&open=0&res=25&view=simple&phrase=1&column=def (pristup ostvaren 27. lipnja 2017.)

[5] http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf (pristup zadnji put ostvaren 19. rujna 2017.)

[6] https://wiki.eprolabs.com/index.php?title=Arduino_Nano (pristup ostvaren 7. lipnja 2017.)

SAŽETAK

Arduinom generirane frekvencije od 10000 Hz do 500000 Hz (s korakom koji u početku kreće od oko 10000 Hz, povećava se s porastom frekvencija (postaje „grublji“) te se pri visokim frekvencijama korak produljuje i to 30000 Hz) koje su željene na ulazu u sklop moraju se dobiti dijeljenjem frekvencije od 16000000 Hz s potrebnim brojem kako bismo dobili željenu frekvenciju uzimajući u obzir da je Duty Cycle 50%, odnosno da u pravokutnom signalu jednako traju high i low (logička ‘1’ i ‘0’) stanje, stoga moramo frekvenciju još podijeliti s 2.
Programobilan mikrokontroler je programiran u “Arduino IDE” programu na način da za zadane frekvencije mjeri napon na sklopu i vrti tu petlju sve dok ne dođe do zadnje frekvencije.
U zadnjem dijelu programa odabire se najpovoljnija frekvencija, tj. ona koja najviše utječe na promjenu napona.

Ključne riječi: Frekvencija, napon, osciloskop, arduino, kapacitivni senzor, pravokutni signal

Autonomus tuning of excitation frequency for capacitive position sensor

ABSTRACT

The Arduin generated frequencies from 10000 Hz to 500000 with a step of 10000 Hz, which are required at the input of the circuit must be obtained by division of frequency of 16000000 Hz with the required number so we can get wanted frequency. We have to pay attention that Duty Cycle is 50%. It means that in the rectangular signal both the high and the low signal state are equal (logical ‘1’ and ‘0’), so we have to divide frequency with 2. The programmable microcontroller is programmed in the “Arduino IDE” program by measuring the voltage in the circuit for the default frequencies and rotetes that loop until the last frequency is reached. The last part of the program selects the most favorable frequncy, actually, the one that most influences the voltage change.

Key words: frequency, voltage, oscilloscope, arduino, capacitive sensor, rectangular signal

ŽIVOTOPIS

Vladimir Mičić rođen je 17.11.1995. godine. U Kneževim Vinogradima završava „Osnovnu školu Kneževi Vinogradi“, nakon čega upisuje “Gimnaziju Beli Manastir”, opći smjer.
Nakon srednjoškolskog obrazovanja upisuje preddiplomski studij elektrotehnike na “Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek”. Na početku 2. godine studija opredjeljuje se za smjer komunikacije i informatka.

Tijekom školovanja sudjelovao je na županijskom natjecanju iz filozofije, a na školskom iz informatike i povijesti.

U Osijeku, lipanj 2017. Vladimir Mičić

_____

(vlastoručni potpis)

PRILOZI

Prilog 1 – Kod pisan u programu “Arduino IDE”

```
// djelitelji za postavljanje frekvencije
long  divisor[] = {800, 400, 267, 200, 160, 133, 114, 100, 89, 80,
                   73, 67, 62, 57, 53, 50, 47, 44, 42, 40,
                   38, 36, 35, 33, 32, 31, 30, 29, 28, 27,
                   26, 25, 24, 23, 22, 21, 20, 19, 18, 17,
                   16
                  };

// vrijednosti frekvencije
double  freq[] = {10000, 20000, 29962.55, 40000, 50000, 60150.38, 70175.44,
                  80000, 89887.64, 100000, 109589.04, 119402.99, 129032.26,
                  140350.88, 150943.40, 160000, 170212.77, 181818.18, 190476.19,
                  200000, 210526.32, 222222.22, 228571.43, 242424.24, 250000,
                  258064.52, 266666.67, 275862.07, 285714.29, 296296.30, 307692.31,
                  320000, 333333.33, 347826.09, 363636.36, 380952.38, 400000,
                  421052.63, 444444.44, 470588.24, 500000
                 };

// varijable potrebne za manipulaciju podacima
unsigned long previousMillis = 0;
const long interval = 100;// interval promjene frekvencije
int i = 0;                // brojac redoslijeda frekvencija
const int sensorPin = A0; // pin za mjerenje napona
int sensorValue = 0;      // vrijednost izmjerena na pinu A0 (napona) 0-1023
float voltage = 0;        // vrijednost napona u decimalnom obliku 0-5V
int napon[41];        // niz izmjerenih napona
float absolute[41];   // niz razlika između prosječnog napona i svih napona
int najveci=-1;   // niz razlika između prosječnog napona i svih napona
int najmanji=1024;   // najmanja razlika između prosječnog napona i svih napona
int minJ = -1;      // mjesto u nizu u kojem je najmanji napon
```

```
float minDelta = 5;   // najmanja razlika između prosječnog napona i svih napona
int gotovo = 0;       // ako su sve frekvencije testirane gotovo = 1

void setup() {
  // početak serijske komunikacije
  Serial.begin(9600);
  //Onemogući prekide
  cli();
  // Izbriši vrijednosti iz registara Timera1
  TCCR1A = 0;
  TCCR1B = 0;
  // Postavi OCR1A (krajnju vrijednost): 16MHz/10KHz/2 = 800 koraka
  OCR1A = 800;
  // COM1A0 za promjenu logičkog stanja pina OC1A (pin9)
  // kad se dosegne krajnja vrijednost
  TCCR1A = _BV(COM1A0);
  // WGM12 za CTC način rada
  // CS10 za postavljanje prescalera 1
  TCCR1B = _BV(WGM12) | _BV(CS10);
  // Postavljanje OC1A (Pin 9/Port B Pin 1) kao izlaz
  DDRB |= _BV(1);
}

void moveUp() {            //funkcija za povećanje frekvencije
  i++;                     // povecaj brojač
  if (i > 41)
  {
    stopCounter();         // postoji 41 zadana frekvencija - zaustavi brojač
  }
  else {
    OCR1A = divisor[i];    // postavi novu krajnju vrijednost brojaca (timera)
  }
}
```

```
void stopCounter() {
  TCCR0B &= 0B00101000;   // zaustavi brojač - bitovi CS10, CS11 i CS12 su 0
  int naponi = 0;
  for (int k = 0; k < 41; k++)
  {
   if (napon[k]>najveci){
    najveci=napon[k];     // određivanje najvećeg napona
    }
    else if (napon[k]<najmanji){
    najmanji=napon[k];    // određivanje najmanjeg napona
    }
   }
  // izračun prosječne vrijednosti napona i pretvaranje u vrijednost između 0V i 5V
  double prosjek = (najveci+najmanji)/2;
  double prosjekIspis = prosjek * 0.0048828125;
  // ispis prosječnog napona u Serial monitor
  Serial.print("Prosjecni napon: ");
  Serial.print(prosjekIspis);



  Serial.println("V");
  // izračun razlika između prosječnog napona i izmjerenih napona
  for (int j = 0; j < 41; j++)
  {
   float razlika = prosjek - napon[j];
    absolute[j] = abs(razlika);
    // pronalazak najmanje razlike napona
    if (minDelta > absolute[j]) {
      minDelta = absolute[j];
      // mjesto u nizu na kojem je najmanja razlika napona
      minJ = j;
    }
  }
  // završen odabir frekvencije
```

```
  gotovo = 1;
  // podesi odabranu frekvenciju
  OCR1A = divisor[minJ];
  // ispis u Serial monitor
  Serial.print("Odabrana frekvencija: ");
  Serial.print(freq[minJ]);
  Serial.println("Hz");
  TCCR1B =_BV(WGM12)|_BV(CS10); // ponovno pokretanje Timera1

}

void loop() {                 // dio koda koji se stalno izvodi
  if (gotovo != 1) {          // ako još nije odabrana krajnja frekvencija
    unsigned long currentMillis = millis(); // trenutno vrijeme
    // ako je razlika trenutnog i prošlog vremena veća od postavljenog intervala
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
      moveUp(); // pokretanje funkcije za promjenu frekvencije
    }
    // očitavanje vrijednosti na analognom pinu 0,
    // očitavanje napona; vrijednosti od 0 do 1023;
    // 10-bitni ADC (analogno digitalni konverter)
    sensorValue = analogRead(sensorPin);
    voltage = sensorValue * 0.0048828125; // pretvori očitanu vrijednost u napon 0-5V
    napon[i] = sensorValue; // spremanje očitanog napona u niz
    // ispis u Serial monitor
    Serial.print("Napon: ");
    Serial.println(voltage);
    Serial.print("Frekvencija: ");
    Serial.println(freq[i]);
  }
}
```