Vilniaus universiteto
Fizikos fakulteto
Taikomosios elektrodinamikos ir telekomunikacijų institutas

Edgardas Rinkevičius

ELEKTROS ENERGIJOS MATUOKLIS, SKIRTAS IŠMANIESIEMS NAMAMS

Magistrantūros studijų baigiamasis darbas

Telekomunikacijų fizikos ir elektronikos
studijų programa

Studentas	Edgardas Rinkevičius
Leista ginti	2019-05-24
Darbo vadovas	doc. Vytautas Jonkus
Recenzentas	doc. Česlovas Pavasaris
Instituto direktorius	prof. Jonas Matukas

Vilnius 2019

Turinys

Įvadas

Pastaruoju metu, didėjant elektros energijos paklausai ir kainai, vis labiau kyla susidomėjimas suvartojamos elektros energijos efektyvumo didinimu. Norint tai pasiekti, pirmiausia reikia pradėti stebėti kasdien suvartojamos energijos kiekį. Tačiau, dauguma namų ūkių vis dar naudoja tradicinius elektros energijos skaitiklius, kurie nesuteikia jokios informacijos apie gyventojų kasdienį energijos suvartojimą. Elektros energijos skaitiklis, skirtas išmaniesiems namams pašalina šį trūkumą, kadangi gali suteikti informaciją apie kas valandą suvartotą energiją. Tai leidžia žmonėms keisti elektros energijos vartojimo įpročius, dėl ko gali būti sumažintas elektros energijos suvartojimas.

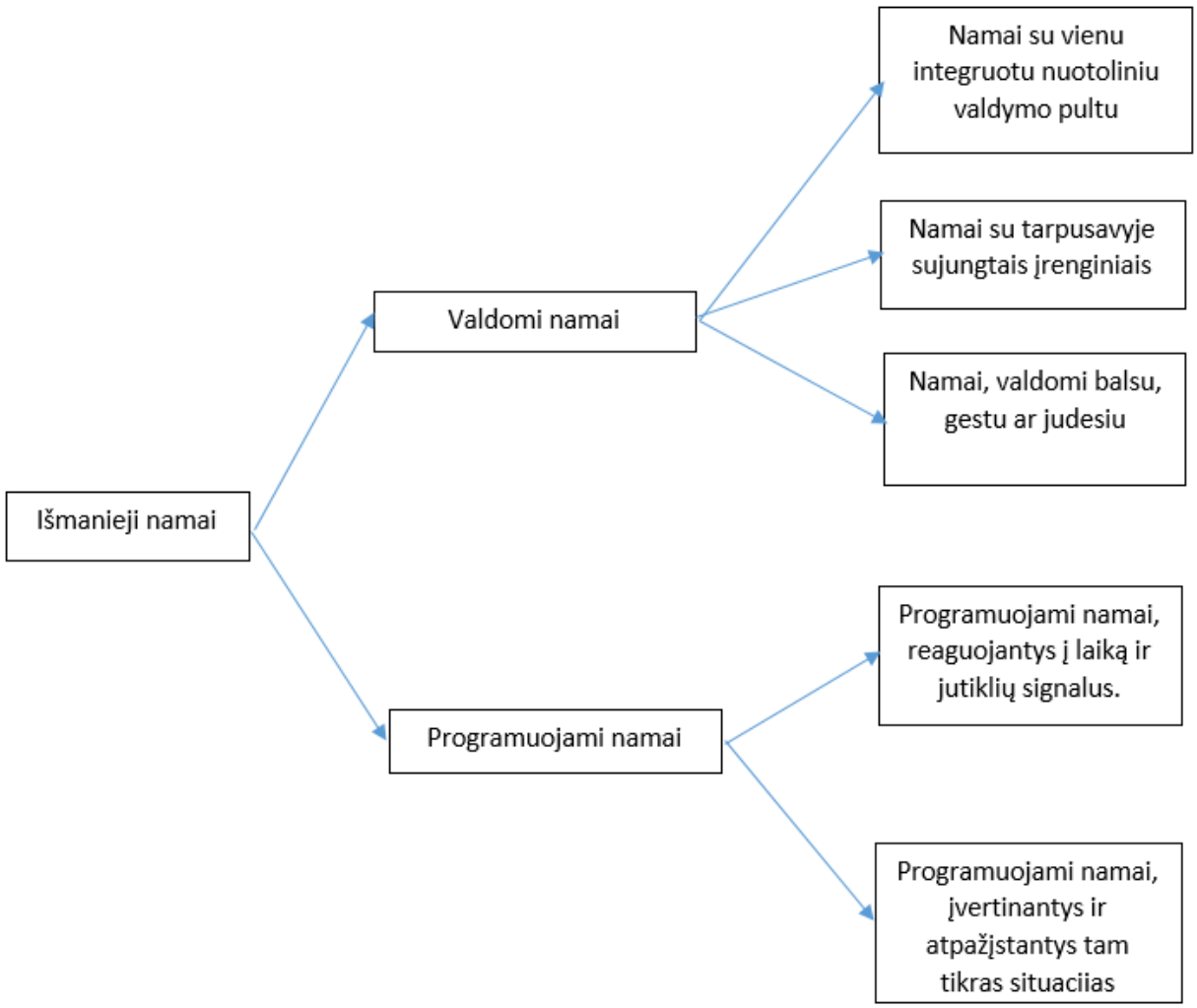
Šiame darbe trumpai aptarti išmanieji namai, apžvelgti elektros energijos matavimo principai bei aptarti elektros energijos srovės bei įtampos efektinių verčių, aktyviosios galios, aktyviosios energijos, reaktyviosios galios ir reaktyviosios energijos skaičiavimo principai. Darbe buvo suprojektuota principinė elektros energijos matavimo prietaiso, skirto išmaniesiems namams, schema bei pagaminta PCB plokštė. Šis matavimo prietaisas realizuojamas aukšto tikslumo elektros energijos matavimo integrinio grandyno ADE7953, kuris gali kaupti didelius kiekius elektros energijos matavimų informacijos. Šis prietaisas matuoja aktyviąją, reaktyviąją ir pilnutinę galią, energiją, o taip pat ir efektines srovės stiprio ir įtampos vertes. ADE7953 sujungtas su mikroprocesoriumi LPC1549, kuris apdoroja duomenis, gautus iš ADE7953 bei pateikia juos LCD ekrane. Sukonstruotas prietaisas buvo užprogramuotas naudojant C programavimo kalbą, tuomet pratestuotas elektros energijos skaitiklių gamybos ir išmaniosios apskaitos sprendimų įmonėje „Elgama-Elektronika“.

Šio darbo tikslas – sukonstruoti elektros energijos matavimo prietaisą, skirtą išmaniesiems namams, tuomet jį sukalibruoti bei ištirti.

1. Literatūros apžvalga

1.1 Išmanieji namai

Išmaniųjų namų technologijos gali reikšti skirtingus dalykus skirtingose aplinkose. Paprastas, bet gudrus jungiklių, valdančių pagrindines namo posistemes (šildymo, apšvietimo ar oro kondicionavimo) ar elektroninius prietaisus, išdėstymas yra puikus pavyzdys technologijų, vadinamų išmaniomis. Taip pat namai, taikantys dirbtinį intelektą tam, kad padėtų žmonėms, gyvenantiems namuose, gali būti priskiriami išmaniųjų namų kategorijai. Kadangi tikslaus išmaniųjų namų sąvokos apibrėžimo nėra, tokie namai gali būti skirstomi į kelias klases. [8]



1 pav. Išmaniųjų namų kategorijos

Kaip matome 1 paveiksle, išmanieji namai gali būti skirstomi į dvi pagrindines kategorijas:

- **Valdomi namai** yra pirmoji kategorija. Tokiame namų tipe tobulinama įvairi įranga, kuria valdomas namas. Tai namai, kuriuose gyventojas gali valdyti įvairias namų sistemas ir

įrenginius patogiau ir efektyviau, negu tai darom paprastuose namuose. Šią kategoriją galima suskirstyti į tris konkrečias klases:

- **Namai su vienu integruotu nuotoliniu valdymo pultu.** Tokiame name kelios posistemės ar prietaisai gali būti valdomi nuotoliniu valdymo pultu ar valdymo skydeliu. Tokiai komunikacijai reikia įdiegti nuotolinio valdymo sąsają tarp nuotolinio valdymo pulto ir įrenginių/posistemių namuose.
- **Namai su tarpusavyje sujungtais įrenginiais.** Įvairūs elektros prietaisai tokie kaip kompiuteriai, televizoriai, elektros energijos suvartojimo stebėjimo prietaisai ir pan. yra sujungti tarpusavyje. Tokia infrastruktūra leidžia keistis informacija tarp įvairių įrenginių. Šiuo atveju namuose reikalingas internetas bei prieš tai aptarto namo tipo funkcijos.
- **Namai, valdomi balsu, gestu ar judesiu.** Ši klasė yra panaši į pirmąją. Esminis skirtumas – nuotolinis valdymo prietaisas yra pakeistas nematomu valdymo bloku, kuris reaguoja į žmogaus balsą ar judesius.

- **Programuojami namai** – antroji išmaniųjų namų kategorija. Tokia infrastruktūra leidžia namą užprogramuoti taip, kad jis galėtų įsijungti, išsijungti ar sureguliuoti kai kuriuos įrenginius esant tam tikroms sąlygoms.
  - **Programuojami namai, reaguojantys į laiką ar jutiklių signalus.** Namai gali įjungti arba išjungti elektros prietaisus esant tam tikram paros metui, pvz. įjungti arba išjungti prietaisus naktį ar švintant. Kitas pavyzdys būtų termostatas, įsijungiantis arba išsijungiantis temperatūrai kokiame nors kambaryje pasiekus reikiamą temperatūrą.
  - **Programuojami namai, įvertinantys ir atpažįstantys tam tikras situacijas.** Tokie namai gali atpažinti vienu metu gaunamus signalus kaip tam tikrus scenarijus. Pavyzdžiui gyventojas, grįžęs po sunkios darbo dienos atsigula ant sofos nusnausti. Tada namas galėtų išjungti šviesas ir įjungti raminamą muziką. Tokie ir panašūs scenarijai gali būti atpažinti, tik jie turi būti iš anksto užprogramuoti. [8, 14]

1.2 Elektros energijos apskaitos svarba

Elektros energijos kainodara gali motyvuoti vartotojus mažinti elektros energijos suvartojimą. Norint mažinti elektros energijos sąskaitas, vartotojai turėtų geriau suprasti kada ir kur energija yra suvartojama, o tai ir yra svarbiausias išmaniųjų elektros energijos skaitiklių pranašumas. Informacija, gauta iš tokių prietaisų gali apriboti energijos nuostolius. Tokių skaitiklių ekranuose gali būti rodoma realiu laiku suvartojamos energijos kainos, o tai leidžia vartotojams planuoti energijos suvartojimą bei keisti energijos vartojimo įpročius. Informacija gali būti siunčiama vartotojui SMS žinute, skaitiklis gali būti jungiamas į bevielį tinklą, prie skaitiklio prijungus „Bluetooth“ anteną.

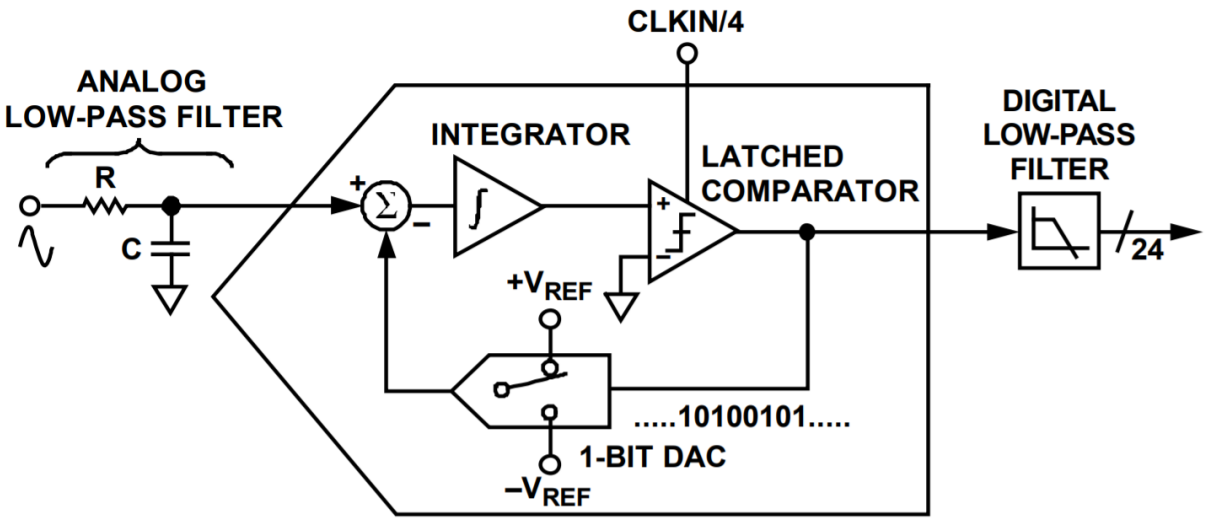
1.3 ADE7953 Elektros energijos matavimo principai

Analoginiai įėjimai.

ADE7953 yra didelio tikslumo elektros energijos matavimo integrinis grandynas, naudojamas vienfaziams matavimams. Šis integrinis grandynas matuoja srovę bei įtampą linijoje ir apskaičiuoja aktyviąją, reaktyviąją ir pilnutinę galią, o taip pat ir efektinę įtampą bei srovę. ADE7953 turi tris analoginius įėjimus, kurie suformuoja du srovės kanalus ir vieną įtampos kanalą. Standartinėje konfigūracijoje, srovės kanalas A matuoja fazinę srovę, o srovės kanalas B matuoja neutraliąją srovę. Įtampos įėjimo kanale matuojamas skirtumas tarp fazės įtampos ir neutralios įtampos. Kanalas A yra sudarytas iš dviejų kojų – IAP ir IAN (5 ir 6 kojos), o kanalas B – iš IBP ir IBN (9 ir 10 kojos). Abu šie kanalai yra pilnai diferencialiniai įtampos įėjimai, naudojami su srovės jutikliu. Įtampos kanalas sudarytas iš VP ir VN (10 ir 11) kojų.[5]

Analogas-skaičius keitiklis

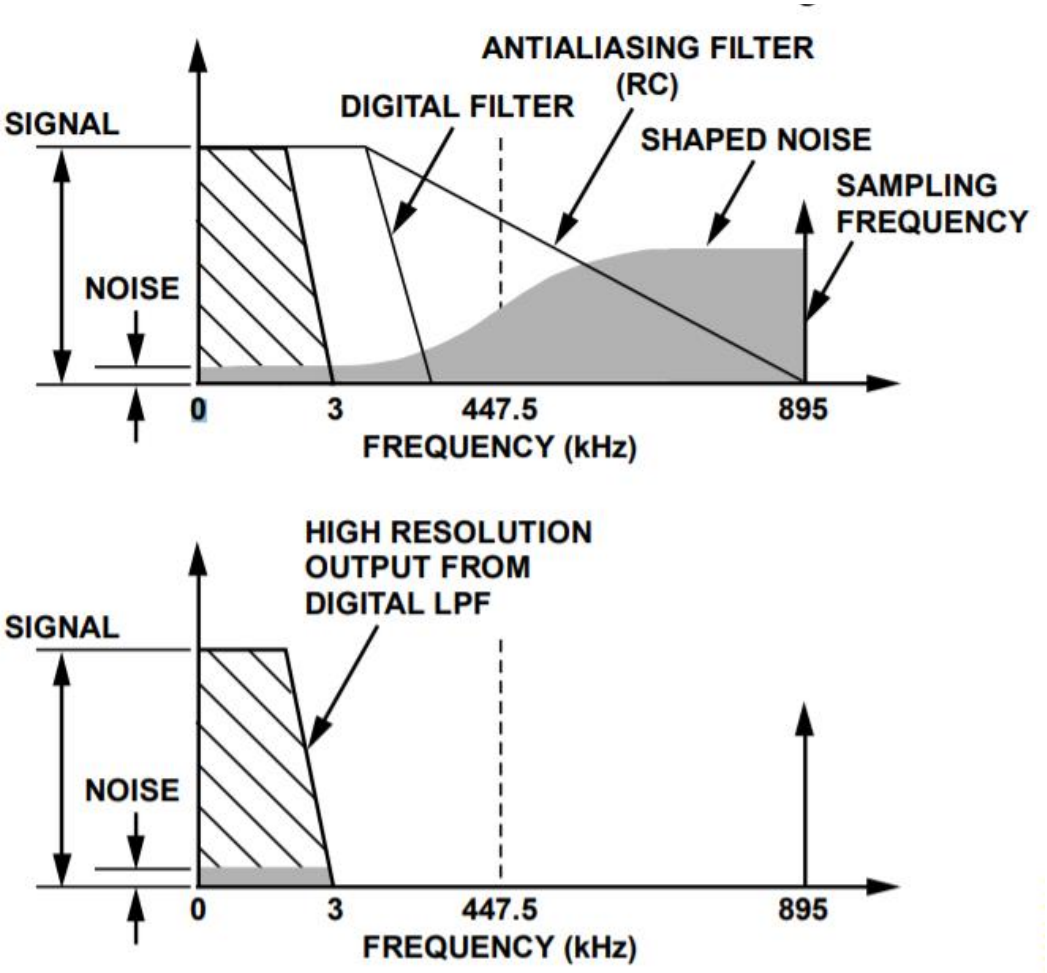
Analoginio signalo keitimas skaitmeniniu ADE7953 yra atliekamas trimis analogas-skaičius moduliatoriais. Moduliatoriaus pavyzdys pateiktas 2 paveikslėlyje.



2 pav. Analoginio signalo keitimas skaitmeniniu [5]

Analogas-skaičius keitiklis, įėjimo signalą paverčia nepertraukiamu vienetų ir nulių srautu, dažniu, nustatomu diskretizavimo laikrodžiu. ADE7953 diskretizavimo dažnis yra lygus 895 kHz (CLKIN/4). Keitiklio veikimo principas realizuojamas pritaikius grįžtamąjį ryšį. Signalas keitiklio išėjime yra atimamas iš keitiklio įėjimo. Jeigu stiprinimas yra pakankamai didelis, vidutinė vertė keitiklio išėjime bus artima vertei keitiklio įėjime. Bet kokiai įėjimo vertei viename diskretizavimo intervale, informacija iš 1-bit ADC yra faktiškai nereikšminga. Reikšmingas rezultatas gaunamas tik suvidurkinus didelį kiekį diskretinių verčių. Vidurkinimo procesas yra atliekamas antroje keitiklio

dalyje – skaitmeniniame žemų dažnių filtre. Suvidurkinus didelį bitų skaičių, skaitmeninis žemų dažnių filtras gali sudaryti 24-bitų žodžius, kurie yra proporcingi įėjimo signalo lygiui. Tam, kad pasiektų didelę rezoliuciją, šis keitiklis panaudoja du metodus – perdiskretizavimą (oversampling) ir triukšmo formos manipuliaciją (noise shaping). Triukšmo mažinimas, panaudojant perdiskretizavimą ir triukšmo formos manipuliaciją, pavaizduotas 3 paveikslėlyje. [5,6]



3 pav. Triukšmo mažinimas, panaudojant perdiskretizavimo ir triukšmo formos manipuliacijos metodus [5]

Perdiskretizavimas

Perdiskretizavimas yra pirmasis metodas, naudojamas aukštai signalo rezoliucijai pasiekti. Perdiskretizavimas reiškia, kad signalas yra diskretizuojamas dažniu, daug didesniu negu signalo juostos plotis. Diskretizavimo dažnis ADE7953 yra 895 kHz, tuo tarpu signalo juostos plotis – nuo 40 Hz iki 1.23 kHz. Perdiskretizavimo pagrindinis efektas – kvantavimo triukšmo išplitimas dažnių juostoje. Tokiu būdu triukšmo lygis mus dominančiame intervale sumažėja. Deja, vien perdiskretizavimo neužtenka norint padidinti signalo-triukšmo santykį. Pavyzdžiui, norint padidinti signalo-triukšmo santykį bent 6 dB, diskretizavimo dažnio koeficientas turi būti 4. Taigi, norint

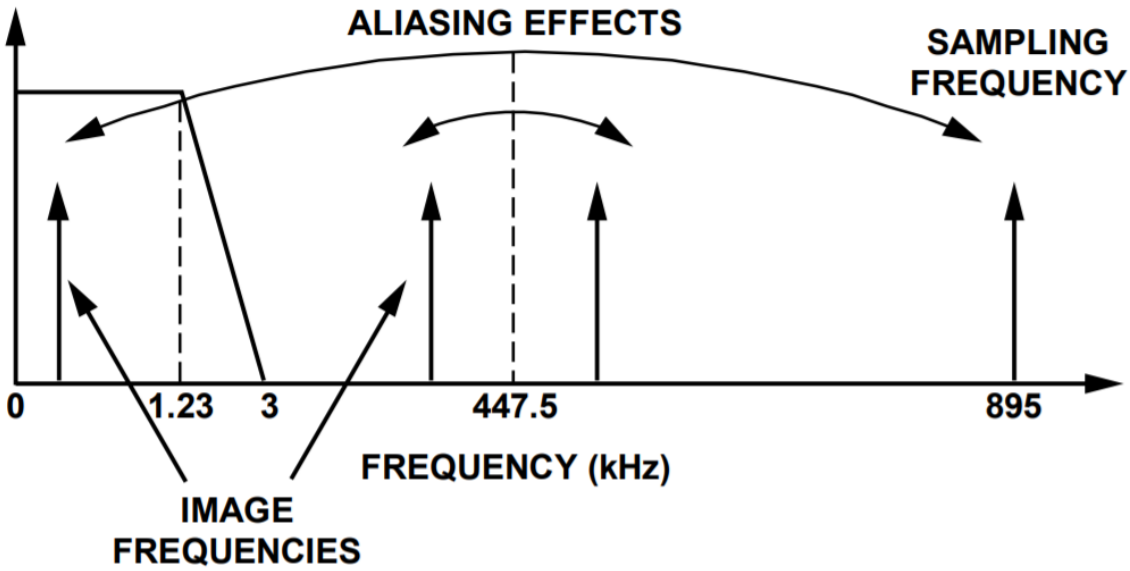
diskretizavimo dažnį išlaikyti priimtiname lygyje, galima manipuliuoti triukšmo forma, tokiu būdu didžioji triukšmo dalis bus aukštesniuose dažniuose, kurie gali būti nufiltruojami. [5,6]

Triukšmo formos manipuliavimas

Antrasis metodas, naudojamas triukšmui sumažinti yra triukšmo formos manipuliavimas. Analogas-skaičius keitiklyje triukšmo forma manipuliuoja integratorius, kuris, dėl grįžtamojo ryšio, turi aukšto dažnio atsaką kvantavimo triukšmui. Dėl to didžioji triukšmo dalis atsiranda ties aukštais dažniais, todėl gali būti nufiltruota skaitmeniniu žemų dažniu filtru, esančiu moduliatoriaus išėjime. [5,6]

Glotninantis filtras

Kaip parodyta 1 paveikslėlyje, prie kiekvieno moduliatoriaus įėjimo reikalingas žemų dažnių RC filtras. Jis reikalingas tam, kad nufiltruotų dažnių komponentes, kurios yra aukštesnės negu pusė diskretizavimo dažnio. Ne esant šiam filtrui, pasireiškia „aliasing“ efektas, kuomet dažnių komponentės, esančios virš pusės diskretizavimo dažnio, atsiranda diskretizuotame signale, kurio dažnis mažesnis negu pusė diskretizavimo dažnio. Šis efektas pavaizduotas 4 paveikslėlyje.

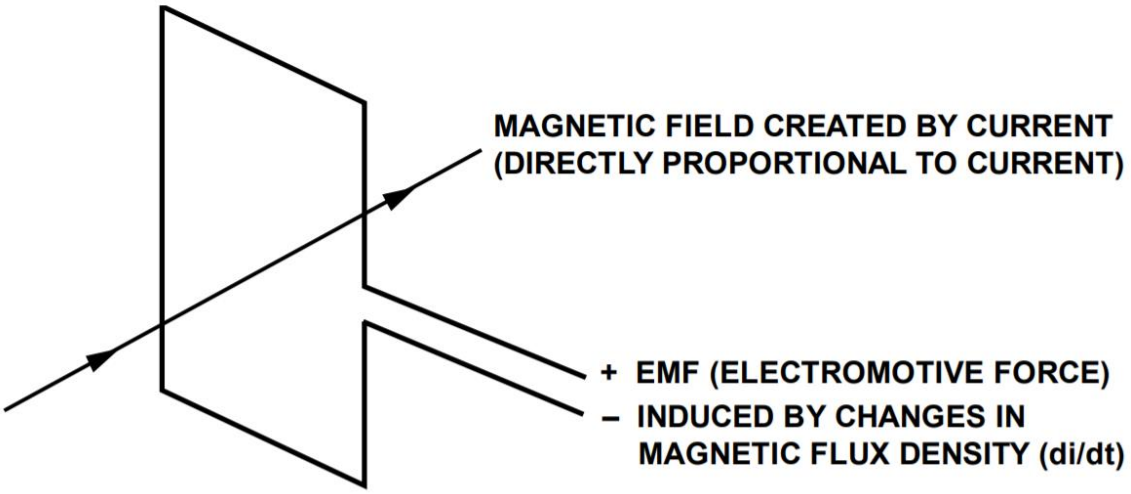


4 pav. „aliasing“ efektas [5]

Iš 4 pav. matome, kad dažnio komponentės, esančios virš Naikvisto dažnio (mūsų atveju 447.5 kHz), atsiranda ir žemiau Naikvisto dažnio. Šis efektas atsiranda visuose analogas-skaičius keitikliuose, nepriklausomai nuo jų architektūros. [5,6]

Srovės sensorius

Srovės sensorius detektuoja magnetinio lauko pokyčius, sukeltus kintamosios srovės. Srovės sensoriaus veikimo principas parodytas 5 paveiksle.



5 pav. Srovės sensoriaus veikimo principas [5]

Magnetinio lauko magnetinė indukcija, indukuota kintamosios srovės, yra tiesiogiai proporcinga tekančios srovės stipriui. Magnetinės indukcijos, kertančios laidininko kilpą, pokytis sukelia elektrovaros jėgą tarp dviejų kilpos galų. Elektrovaros jėga yra įtampos signalas, kuris yra proporcingas srovės pokyčiui laike (di/dt). Įtampa srovės jutiklio išėjime yra apsprendžiama abipusio induktyvumo tarp srovės laidininko ir srovės jutiklio. Jutiklio išėjime turime di/dt signalą, kurį reikia integruoti, norint gauti srovės signalą, todėl jutiklio išėjime turi būti integratorius.[2,5,7]

1.4 Srovės, galios ir energijos matavimai

Efektinės vertės matavimas.

Efektinė vertė yra kintamojo signalo dydžio matavimas. Tiksliau, efektinė kintamojo signalo stiprio ar įtampos vertė yra lygi pastovios srovės ar įtampos dydžiui, reikalingam, norint gauti tokią pačią galią apkrovoje. Srovės efektinė vertė:

$$I_{eff} = \sqrt{\frac{1}{t}\int_0^t I^2(t)dt}, \quad (1)$$

diskretizuotiems signalams, efektinės vertės skaičiavimui reikia harmoninį signalą paversti stačiakampiu signalu, suvidurkinti diskretines vertes ir ištraukti šaknį:

$$I_{eff} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} I^2[n]}, \quad (2)$$

ADE7953 pateikia srovės kanalo A, srovės kanalo B ir įtampos kanalo efektines vertes vienu metu. Šios vertės atnaujinamos 6,99 kHz dažniu. [5,6]

Aktyviosios galios skaičiavimas

Galia apibrėžiama kaip elektros srovės darbas, atliktas per tam tikrą laiko vienetą arba įtampos ir srovės stiprio sandauga. Momentinė srovė ir momentinė galia išreiškiamos taip:

$$U(t) = \sqrt{2} \cdot U_{eff} \cdot \sin(\omega t), \quad (3)$$

$$I(t) = \sqrt{2} \cdot I_{eff} \cdot \sin(\omega t), \quad (4)$$

kur U(t) – momentinė įtampos vertė, U_{eff} – efektinė įtampos vertė, $\omega -$ įtampos kampinis dažnis, I(t) – momentinė srovės vertė, I_{eff} – efektinė srovės vertė, $\omega -$ srovės kampinis dažnis. Tuomet, momentinė galia išreiškiama taip:

$$P(t) = U(t) \cdot I(t), \quad (5)$$

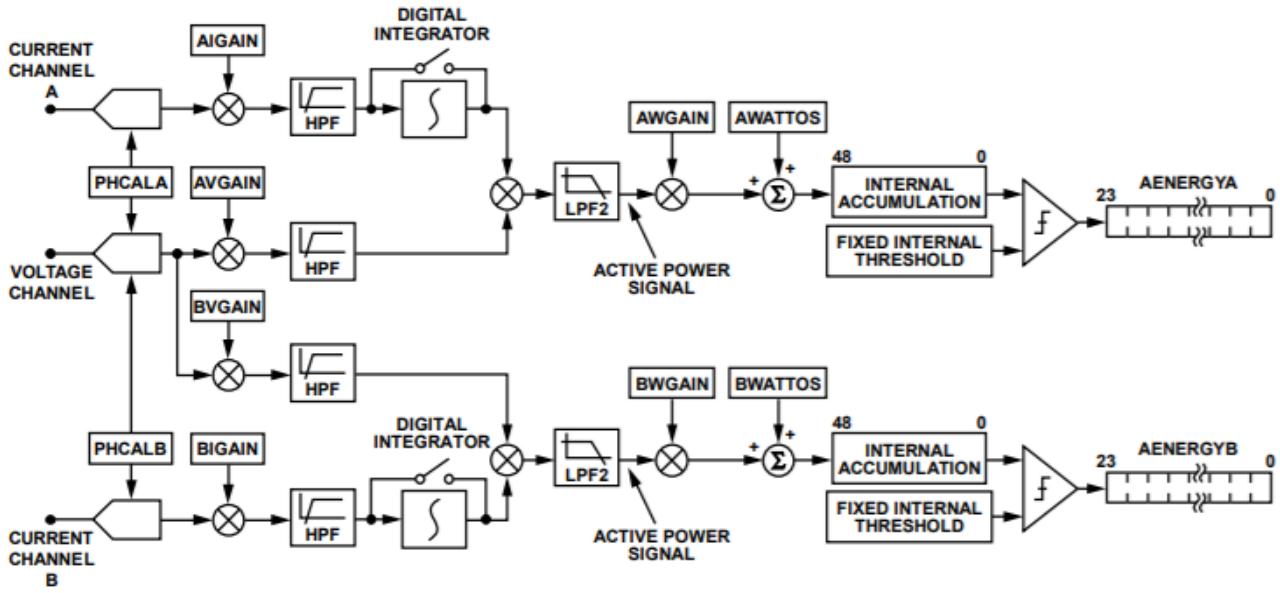
$$P(t) = UI - UIcos(\omega t), \quad (6)$$

čia P(t) – momentinė elektros srovės galia. Vidutinės galios per n periodų išraiška pateikta (7) formulėje:

$$P = \frac{1}{nT}\int_0^{nT} P(t)dt = UI, \quad (7)$$

kur P – aktyvioji galia, T – srovės periodas. Aktyvioji galia yra lygi momentinės galios P(t) pastoviosios srovės komponentei. Reiškia, aktyvioji galia yra lygi įtampos ir srovės stiprio sandaugai.

Grandinė, naudojama galios ir energijos skaičiavimams pavaizduota 6 paveiksle.



6 pav. Elektros energijos ir galios skaičiavimo grandinė [5]

Momentinė galia gaunama sudauginus srovės stiprio ir galios signalus. Tuomet pastovios srovės komponentė yra išgaunama panaudojus LPF2 filtrą – žemų dažnių filtrą, tokiu būdu gaunant informaciją apie aktyviąją galią. ADE7953 aktyviąją galią vienu metu apskaičiuoja ir srovės kanale A ir srovės kanale B. Aktyviosios galios registrai taip pat atnaujinami 6.99 kHz dažniu. [5,6]

Aktyviosios energijos skaičiavimas

Kaip jau minėta anksčiau, galia – tai elektros srovės atliktas darbas per tam tikrą laiko vienetą. Matematiškai ši sąryšis gali būti išreikštas taip:

$$P = \frac{dE}{dt}, \tag{8}$$

kur P – elektros srovės galia, E – elektros srovės energija. Tokiu atveju, elektros energiją galime išreikšti taip:

$$E = \int Pdt, \tag{9}$$

ADE7953 aktyviosios galios integravimą pasiekia dviem etapais. Pirmame etape, aktyviosios galios signalai yra kaupiami vidiniame 48 bitų registre kas 143 µs (6.99 kHz), kol pasiekiamas tam tikras nustatytas slenkstis. Kai šis slenkstis yra pasiekiamas, generuojamas impulsas, kuris kaupiamas 24 bitų vartotojui prieinamuose registruose. Diskretinis impulsų sumavimas yra ekvivalentus nuolatinio signalo integravimui. Šis sąryšis išreikštas (10) formulėje.

$$E = \int Pdt = \lim_{T \to 0} \{\sum_{n=1}^{\infty} P(nT) \cdot T, \tag{10}$$

čia n – diskretizuotų impulsų kiekis, T – diskretizavimo periodas. [5,6]

Reaktyviosios galios skaičiavimas

Reaktyvioji galia – tai srovės stiprio ir įtampos sandauga, kai vieno iš šių dydžių fazė yra pasukta 90º. Gautas signalas vadinamas momentine reaktyviąja galia. Momentinės reaktyviosios galios išraiška, kai srovės kanalas pasuktas +90º pateikta (11) ir (12) lygtyse:

$$P_r(t) = U(t) \cdot I'(t), \quad (11)$$

$$P_r(t) = UI \cdot \sin(\theta) + UI \cdot sin(2\omega t + \theta), \quad (12)$$

$$U(t) = \sqrt{2} \cdot Usin(\omega t + \theta), \quad (13)$$

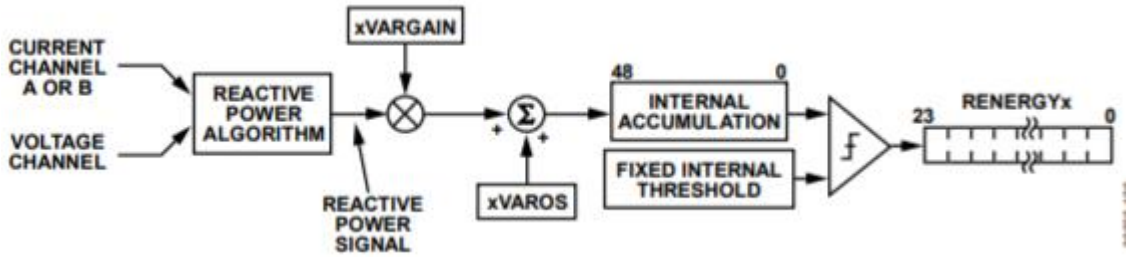
$$I(t) = \sqrt{2} \cdot Isin(\omega t), \quad (14)$$

$$I'(t) = \sqrt{2} \cdot Isin(\omega t + \frac{\pi}{2}), \quad (15)$$

kur $P_r(t)$ – momentinė reaktyvioji galia, U – efektinė įtampa, I - efektinis srovės stipris, θ – fazių skirtumas tarp srovės stiprio ir įtampos. Analogiškai aktyviosios galios skaičiavimams, reaktyvioji galia per tam periodų skaičių n yra lygi:

$$P_r(t) = \frac{1}{nT}\int_0^{nT} P_r(t)dt = UIsin(\theta). \quad (16)$$

Reaktyvioji galia yra lygi momentinės reaktyviosios galios nuolatinės srovės komponentei. Grandinė reaktyviosios galios skaičiavimui pavaizduota 7 paveiksle.



7 pav. reaktyviosios galios ir energijos skaičiavimo grandinė [5]

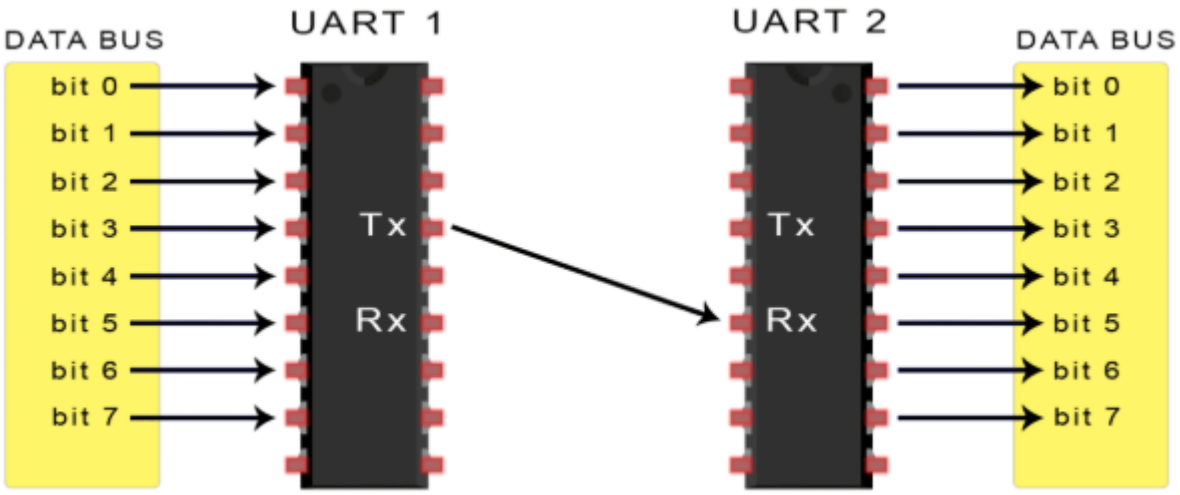
Momentinė reaktyvioji galia gaunama sudauginus srovės signalą su įtampos signalu. Skaičiavimai atliekami srovės kanale A ir srovės kanale B vienu metu. Daugyba atliekama visame 1.23 kHz dažnių juostos plotyje, ko rezultate gauname reaktyviosios galios vertę su visomis harmonikomis, esančiomis toje dažnių juostoje. Tuomet gautą rezultatą praleidžiame per žemų dažnių filtrą ir gauname reaktyviosios galios informaciją.

Reaktyvioji energija gaunama analogiškai aktyviosios galios gavimo principui, tik šiuo atveju integruojama reaktyvioji galia. [2,5]

1.5 UART sąsaja

Šiame elektros energijos matuoklyje, komunikacijai tarp integrinio grandyno ADE7953 ir mikroprocesoriaus LPC1549 panaudota UART sąsaja. UART (*Universal Asynchronous Receiver – Transmiter*) – tai universalus asinchroninis imtuvas-siųstuvas, prietaisas, kurio pagrindinė paskirtis - siųsti ir priimti nuoseklią informaciją. Tai ne komunikacijos protokolas kaip SPI ar I2C, o fizinė

mikrovaldiklio grandinė. Pagrindinis UART sąsajos privalumas – naudojami tik du kanalai informacijos perdavimui tarp siųstuvo ir imtuvo. UART komunikacijoje, du prietaisai tiesiogiai komunikuoja tarpusavyje. UART1 siųstuvas konvertuoja lygiagrečią informaciją iš mikrovaldiklio į nuoseklią formą, siunčia į UART2 imtuvą, kuris konvertuoja priimtą nuoseklią informaciją atgal į lygiagrečią formą. Informacijai perduoti naudojami tik du fiziniai kanalai. UART komunikacija pavaizduota 8 paveiksle.



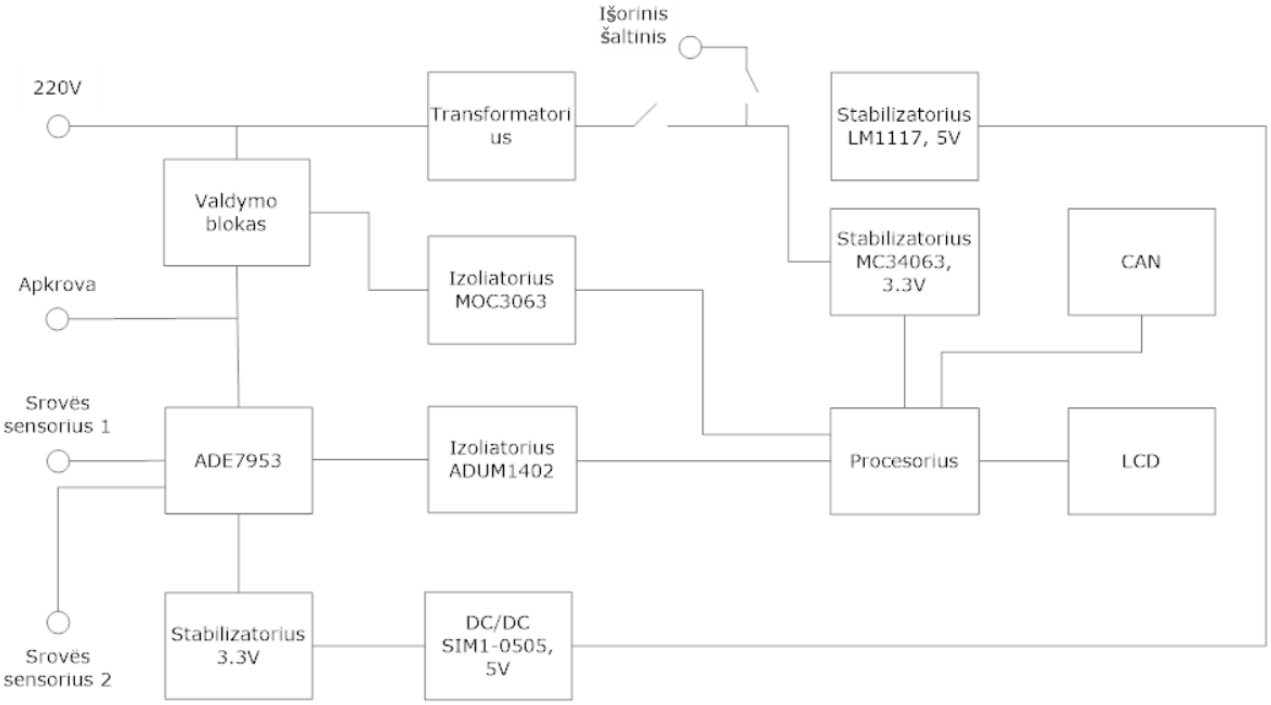
8 pav. UART komunikacija [12]

Informacija UART sąsaja perduodama asinchroniškai, tai reiškia, kad nėra laikrodžio, kuris galėtų sinchronizuoti siųstuvo ir imtuvo darbą. Todėl UART siųstuvas prideda pradžios ir pabaigos bitus, tokiu būdu pažymėdamas duomenų paketo pradžią ir pabaigą [13]

2. Darbo eiga

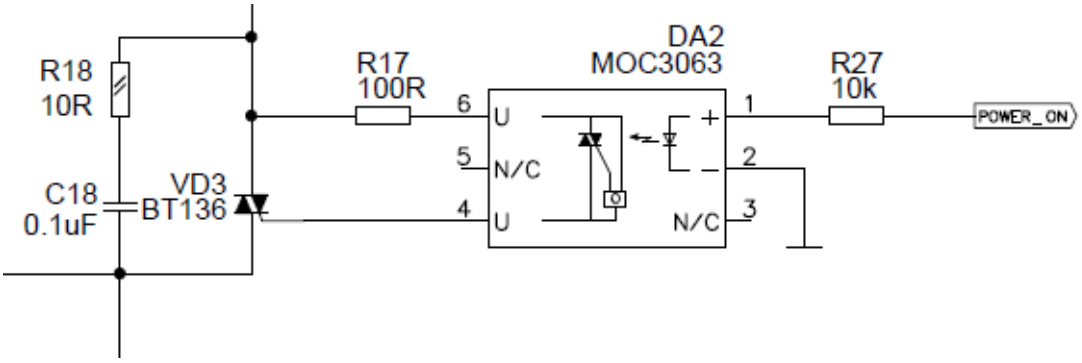
2.1 Principinės schemos projektavimas

Šio darbo metu buvo suprojektuota principinė elektros energijos skaitiklio schema, o taip pat ir skaitiklio PCB plokštė. Suprojektuoto elektros energijos skaitiklio blokinė schema pavaizduota 9 pav.



9 pav. elektros energijos skaitiklio blokinė schema

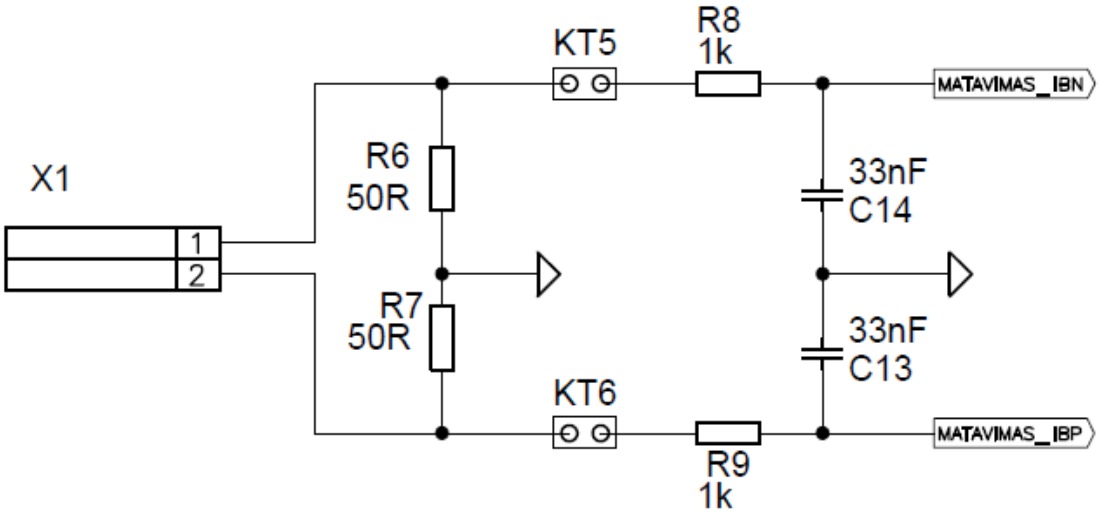
Valdymo blokas ir izoliatorius



10 pav. valdymo bloko principinė schema

Valdymo bloko principinė schema pateikta 10 paveiksle. Pagrindiniai elementai, realizuojantys valdymo bloko veikimą yra optronas MOC3063 ir simistorius VD3. Optronas – tai prietaisas, šviesos pagalba perduodantis signalą tarp dviejų elektriškai izoliuotų grandinių. Perduodant šviesos signalą, siųstuvas yra šviesos diodas, o imtuvas – fotodiodas. Simistorius šioje grandinėje – apkrovą komutuojantis elementas. Elementai R18 ir C18 skirti apsaugoti simistorių nuo aukštos įtampos šuolių, kai simistorius išjungia apkrovą.

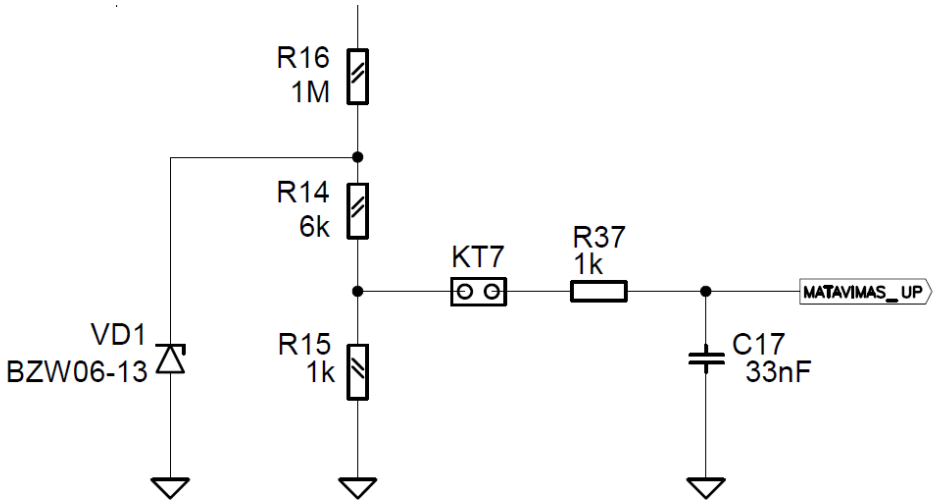
Srovės sensoriai



11 pav. srovės sensoriaus principinė schema

Srovės sensoriaus principinė schema pateikta 11 paveiksle. Srovės jutiklio veikimo principas paremtas elektromagnetine indukcija. Srovės pokytis linijoje bus proporcingas indukuotos srovės pokyčiui matavimo grandinėje. Srovės sensorių realizuojanti schema pavaizduota 11 paveiksle. Elementai R8 ir C14, R9 ir C13 – tai žemų dažnių filtrai, skirti pašalinti aukšta dažniams triukšmams. Rezistoriai R6 ir R7 yra apkrovos rezistoriai. KT5 ir KT6 – jungtys, skirtos prietaiso testavimui ir kalibravimui.

Įtampos matavimo grandinė

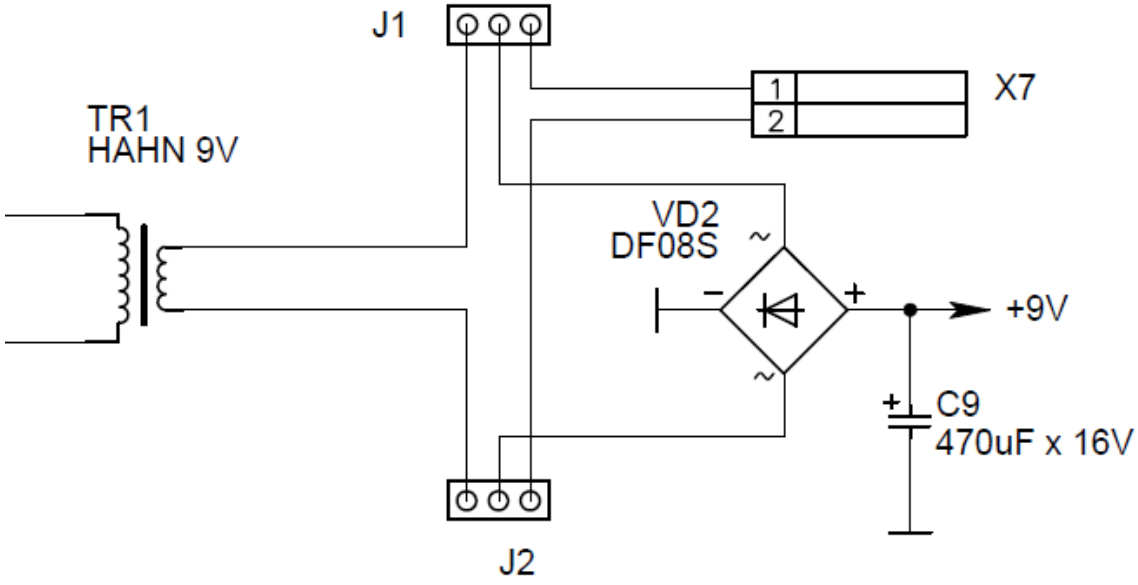


12 Pav. Įtampos matavimo grandinė

Įtampos matavimas šioje schemoje realizuojamas įtampos daliklio pagalba. Rezistoriai R16, R14 ir R15 sudaro įtampos daliklį. VD1 – apsauginis diodas, apsaugantis nuo didelių įtampos šuolių. Šis diodas, esant dideliems įtampos šuoliams (šiuo atveju – daugiau negu 13V), srovę įžemina. Toliau

esantys įtampos dalikliai R14 ir R15 sumažina įtampą iki reikiamos. Rezistorius R37 ir kondensatorius C17 – žemų dažnių filtras.

Transformatorius

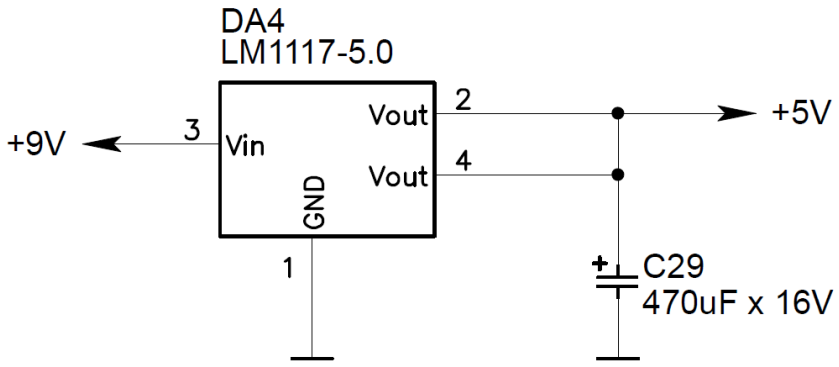


13 pav. transformatorius su srovės lygintuvu

Šioje principinės schemos dalyje transformatorius TR1 skirtas mažinti tinklo kintamąją įtampą nuo 220 V pirminėje vijoje iki 9 V antrinėje vijoje. Antrinė vija prijungta prie srovės lygintuvo DF08S, kuris kintamąją srovę paverčia +9 V nuolatine srove. Jungtys J1 ir J2 atlieka komutatoriaus vaidmenį – srovės lygintuvą sujungia su antrine transformatoriaus vija arba išoriniu įtampos šaltiniu.

Stabilizatoriai

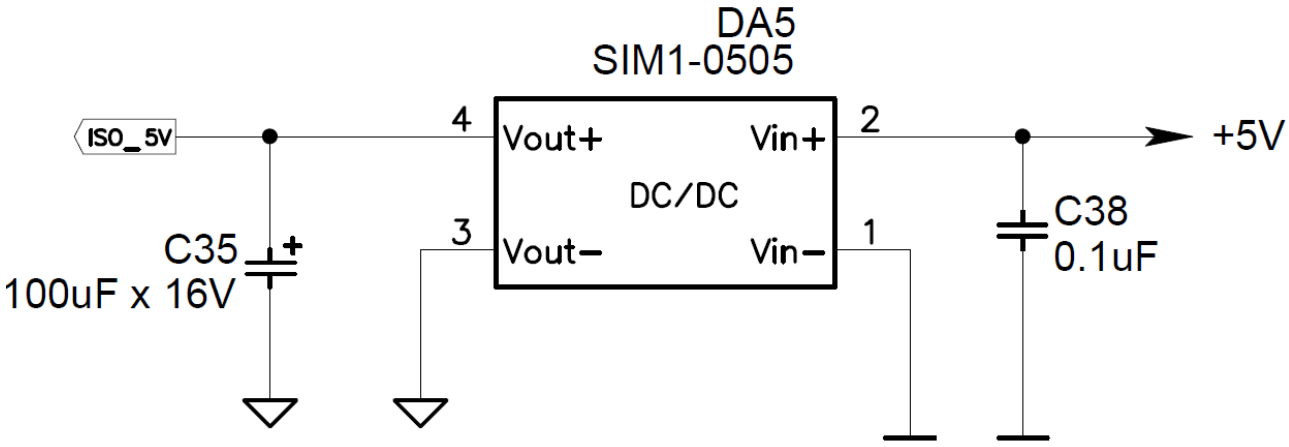
14 pav. stabilizatorius LM1117



15 pav. stabilizatoriaus LM1117 jungimo schema

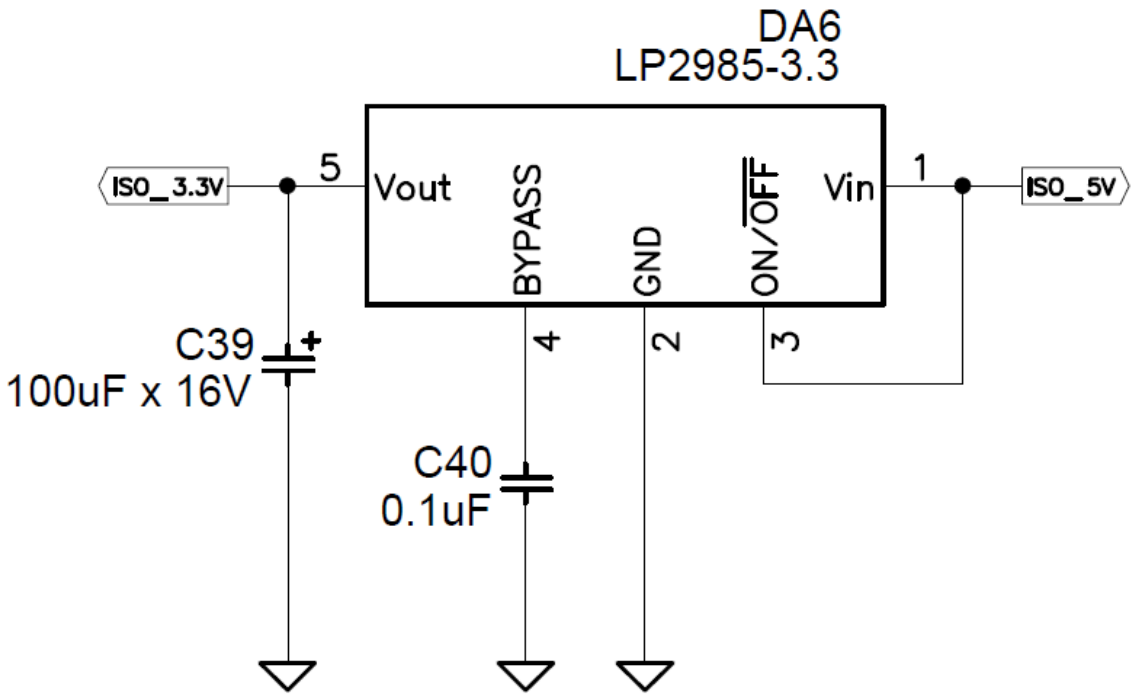
Stabilizatorius LM1117 – tiesinis žeminantysis stabilizatorius, iš srovės lygintuvo paduodamą +9 V įtampą mažinantis iki +5 V. Šia įtampa yra maitinamas LCD ekranas, taip pat įtampa paduodama į DC/DC keitiklį SIM1-0505, kurio jungimo schema pavaizduota 16 paveiksle.

16 Pav. DC/DC keitiklis SIM1-0505



17 Pav. DC/DC keitiklio jungimo schema

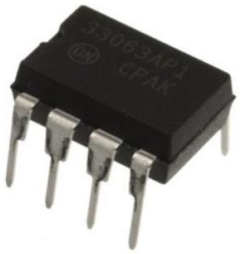
Šis DC/DC keitiklis schemoje panaudotas kaip izoliatorius, norint galvaniškai atskirti integrinį grandyną ADE7953 nuo LCD ekrano maitinimo grandinės. Tuomet ši įtampa yra žeminama tiesiniu žeminančiuoju stabilizatoriumi LP2985, kurio jungimo schema pavaizduota 18 paveiksle.

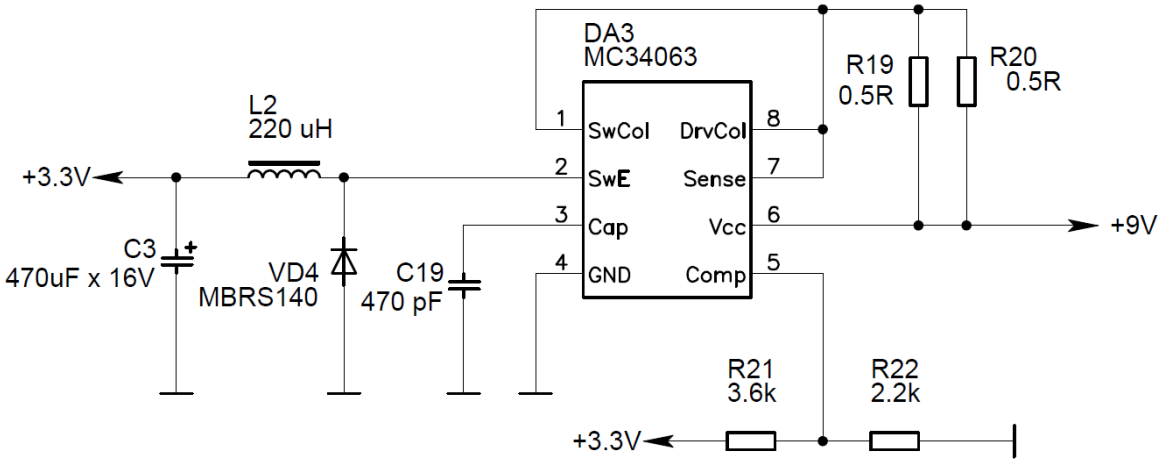


18 pav. stabilizatoriaus LP2985 jungimo schema

Žeminantysis tiesinis stabilizatorius LP2985, +5 V įtampą žemina iki +3.3 V įtampos, kuria maitinamas integrinis grandynas ADE7953.

19 paveiksle pavaizduotas žeminantysis impulsinis stabilizatorius MC34063 ir jo jungimo schema.



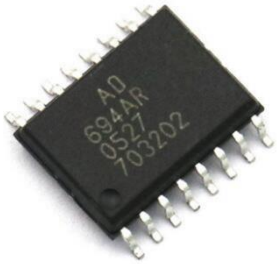


19 pav. stabilizatorius MC34063 ir jo jungimo schema

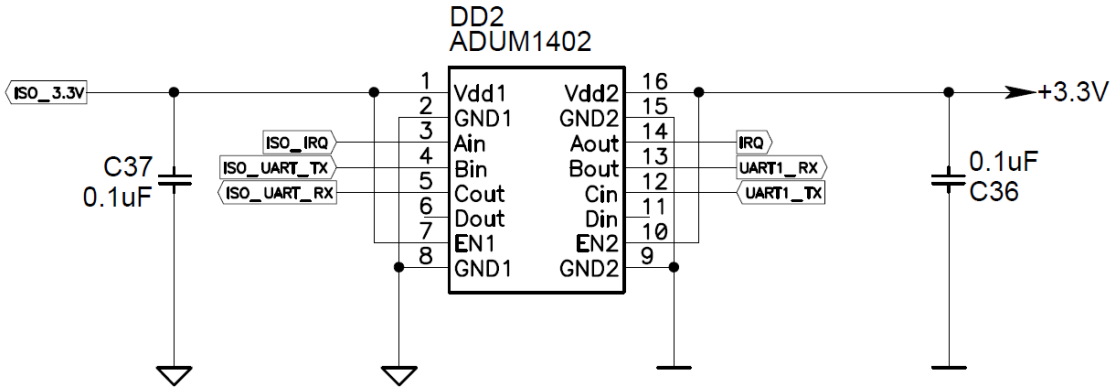
Šioje schemoje stabilizatorius MC34063 yra prijungtas kaip žeminantysis impulsinis stabilizatorius, žeminantis +9 V įtampą, ateinančią iš srovės lygintuvo, iki +3.3 V įtampos, kuria maitinamas procesorius LPC1549. Stabilizatorius MC34063 išėjime (2 koja) siunčia stačiakampio formos impulsus. Ritė L2 ir kondensatorius C3 sudaro LC filtrą, kuris suvidurkina stačiakampius impulsus, tokiu būdu juos paversdamas nuolatine įtampa. Įtampos vertė gali būti reguliuojama, keičiant impulsų trukmę. Rezistoriai R21 ir R22 nustato išėjimo įtampą (3.3 V)

Izoliatorius

20 paveiksle pavaizduotas izoliatorius ADUM-1402 ir jo jungimo schema



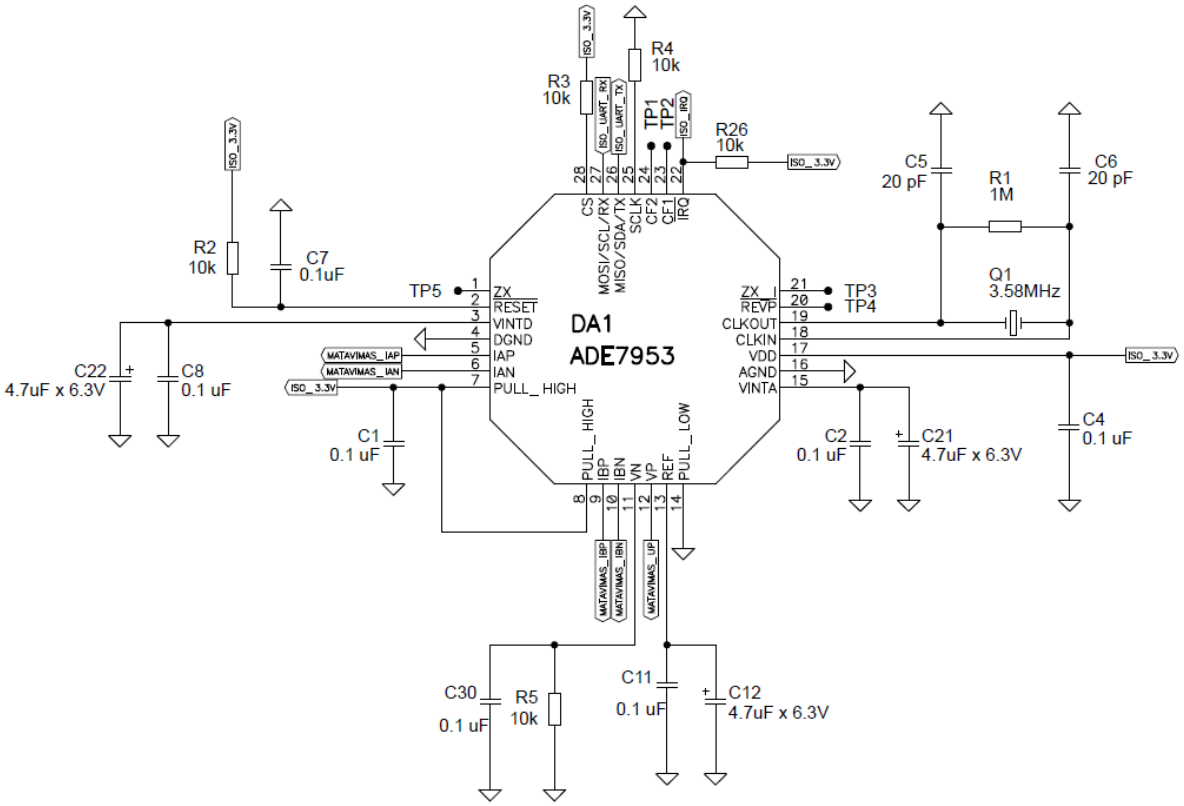
20 pav. Izoliatorius ADUM1402



21 pav. izoliatoriaus ADUM1402 jungimo schema

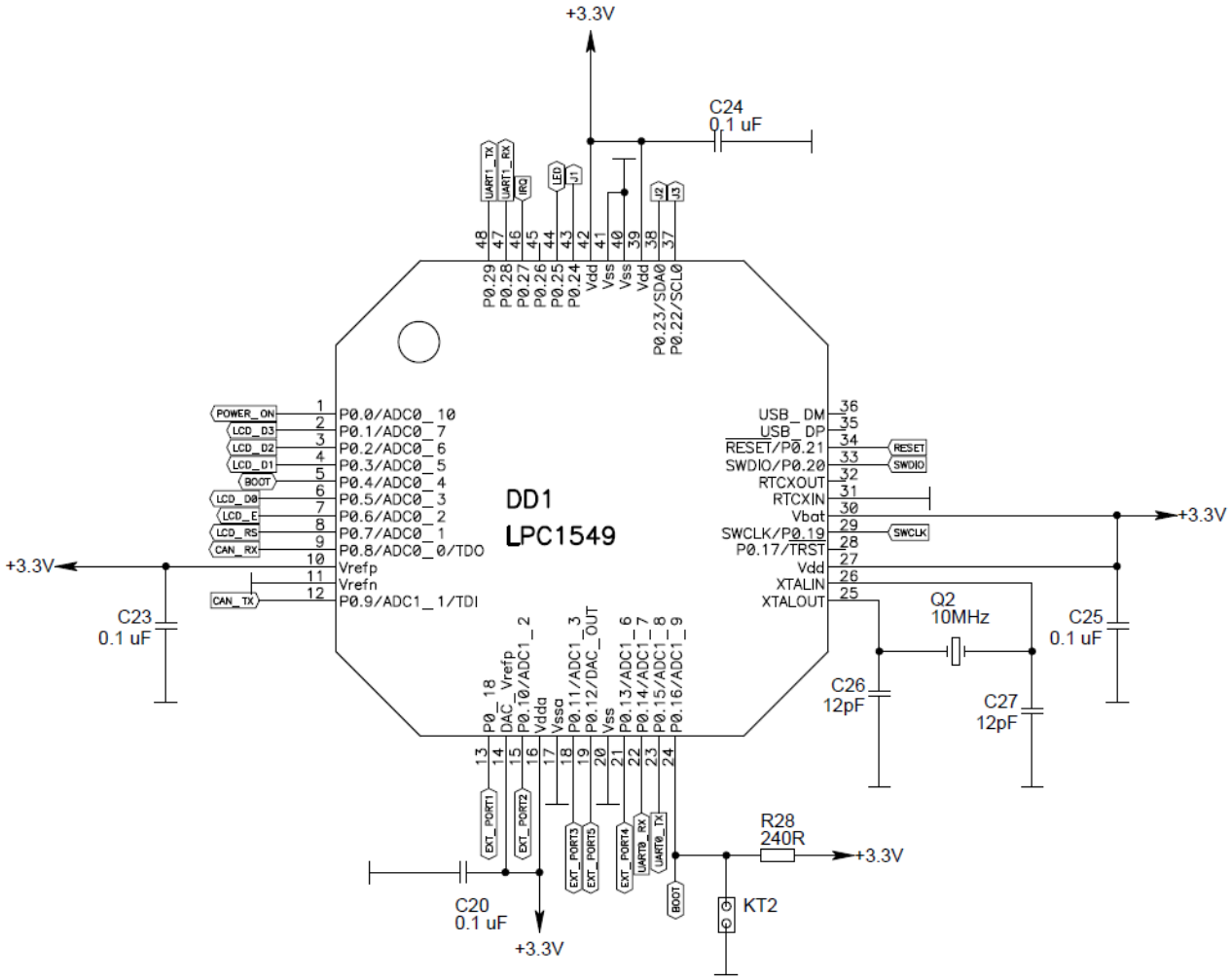
ADUM1402 – skaitmeninis izoliatorius, skirtas galvaniškai izoliuoti integrinio grandyno ADE7953 grandinę nuo procesoriaus grandinės.

ADE7953 ir procesorius LPC1549



22 pav. ADE7953 jungimo schema

ADE7953 – integrinis grandynas, matuojantis srovę ir įtampą linijoje, skaičiuoja aktyviąją, reaktyviąją, pilnutinę energiją, o taip pat – efektinę srovę bei įtampą. Prie ADE7953 CLKIN (18) ir CLKOUT (19) kojų prijungtas kvarcinis rezonatorius Q1, kurio vibravimas naudojamas labai tikslaus dažnio elektriniam signalui generuoti, kuris panaudojamas laiko sekimui. Signalo dažnis – 3.58 MHz. ADE7953 su procesoriumi LPC1549 komunikuoja naudodama UART sąsają. Procesoriaus LPC1549 jungimo schema pavaizduota žemiau pateiktame paveiksle.



23 pav. procesoriaus LPC1549 jungimo schema

LPC1549 yra ARM Cortex-M3 pagrindu sukurtas mikrovaldiklis, skirtas įterptinių sistemų (embedded systems) taikymui. LPC1549 dirba dažniuose iki 72 MHz. Šis mikrovaldiklis turi 256 kB vidinės flash atminties. Prie XTALIN ir XTALOUT kojų prijungto kvarcinio rezonatoriaus Q paskirtis analogiška rezonatoriui Q1, prijungtam prie ADE7953, tik šio rezonatoriaus dažnis – 10 MHz.

2.2 Skaitiklio konstravimas

Skaitiklio konstravimas vyko keliais etapais:

1) Pirmiausia buvo pagaminta PCB plokštė. Plokštės gamybai buvo panaudotas foto-rezistyvinis ėsdinimo metodas. Jame naudoja stiklo tekstolito plokštė, padengta variu, o varis padengtas fotorezisto sluoksniu. Fotorezistas – tai medžiaga, kuri tampa atspari ėsdinimui (cheminiam veikimui) ją apšvietus šviesos spinduliais. Ant fotorezisto sluoksnio uždėjus takelių kaukę (atspausdintą suprojektuotą PCB plokštę) ir ją apšvietus UV spinduliais susidaro apšviestos ir neapšviestos sritys. Tuomet plokštė patalpinama į natrio chlorido tirpalą, kuriame pašalinamos neapšviestos plokštės sritys. Pašalinus fotorezistą, plokštė patalpinama į druskos

tirpalą, kuris nuėsdina atidengtas vario dalis. Tai atlikus pašalinamas likęs fotorezisto sluoksnis, gauname PCB plokštę su suprojektuota schema.

2) Atidengtas vario sluoksnis padengiamas lydmetaliu.
3) PCB plokštėje išgręžiamos skylutės komponentų montavimui, takelių perėjimams iš vieno sluoksnio į kitą.
4) Galiausiai sumontuojami komponentai. Pirmiausia lituojami pasyvieji elementai - rezistoriai, kondensatoriai ir ritės. Tuomet lituojami stabilizatoriai, keitikliai, jungtys. Sulitavus procesoriaus ir integrinio grandyno ADE7953 schemas, galima pradėti programuoti.

2.3 Programavimas

2.3.1 Surašytos procesoriaus LPC1549 ir matavimo integrinio grandyno ADE7953 kojų sąrašai ir jų Adresai. *[1, 2, 3 priedai]*

2.3.2 Parašyta pagrindinė main() programa, joje surašytos funkcijos:

1) Procesoriaus taktinio dažnio skaičiavimui.
2) Procesoriaus palaikomų sąsajų inicializavimui, šviesos diodo valdymas.
3) Parašyta testavimo funkcija, skirta patikrinti komunikacijai tarp procesoriaus ir integrinio grandyno. *[4, 5 priedai]*

2.3.3 Parašytos funkcijos duomenų įrašymui ir nuskaitymui iš integrinio grandyno ADE7953 registrų. *[6 priedas]*

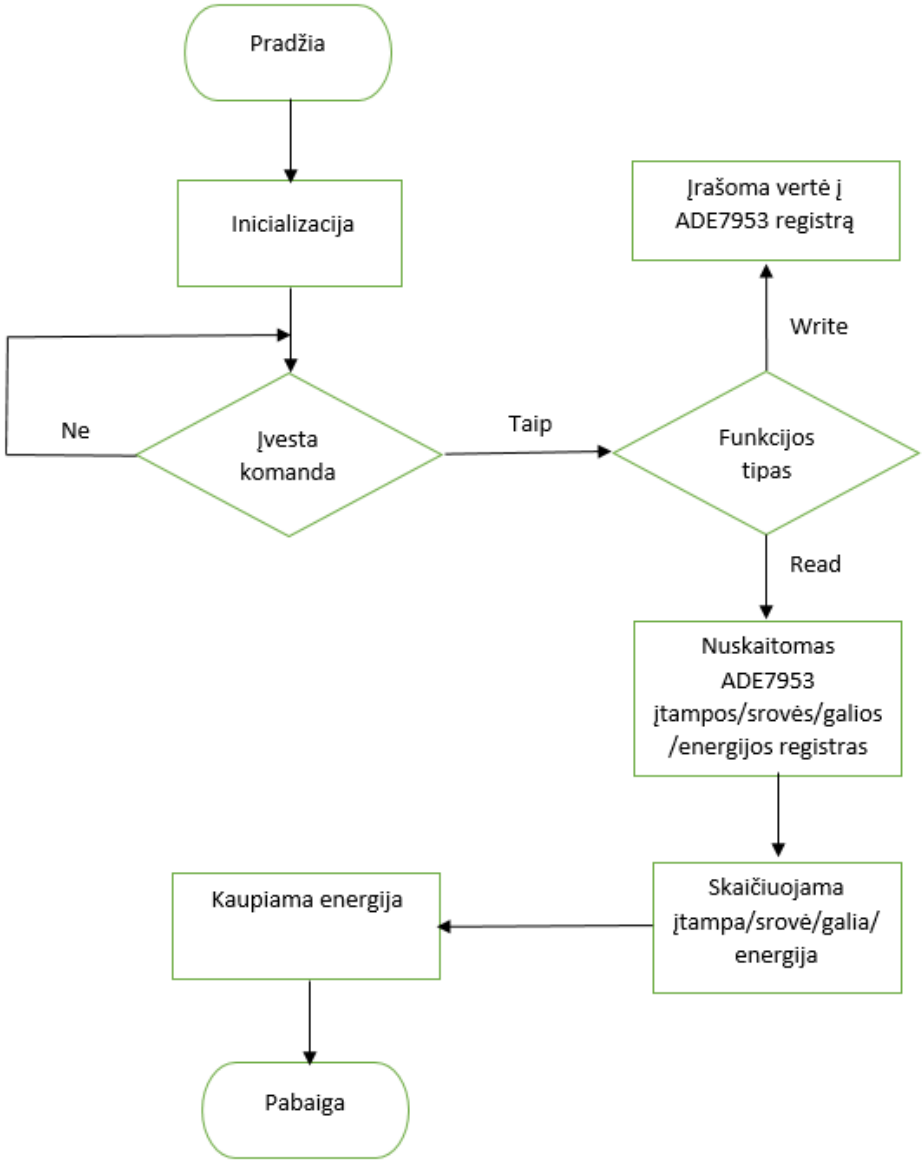
2.3.4 Parašytos funkcijos sukalibruotoms elektros srovės, įtampos, aktyviosios ir reaktyviosios galios ir energijos vertėms matuoti. *[7 priedas]*

2.3.5 Parašytos funkcijos valdymui per komandinę eilutę. *[8 priedas]*

2.3.6 Pataisytas nustatymų failas: įrašyti UART modulio nustatymai, veikimo dažniai. *[9 priedas]*

2.3.7 Parašyta programa sukompiliuota ir įkelta į plokštę.

Parašytos programos blokinė schema pavaizduota 24 paveiksle.

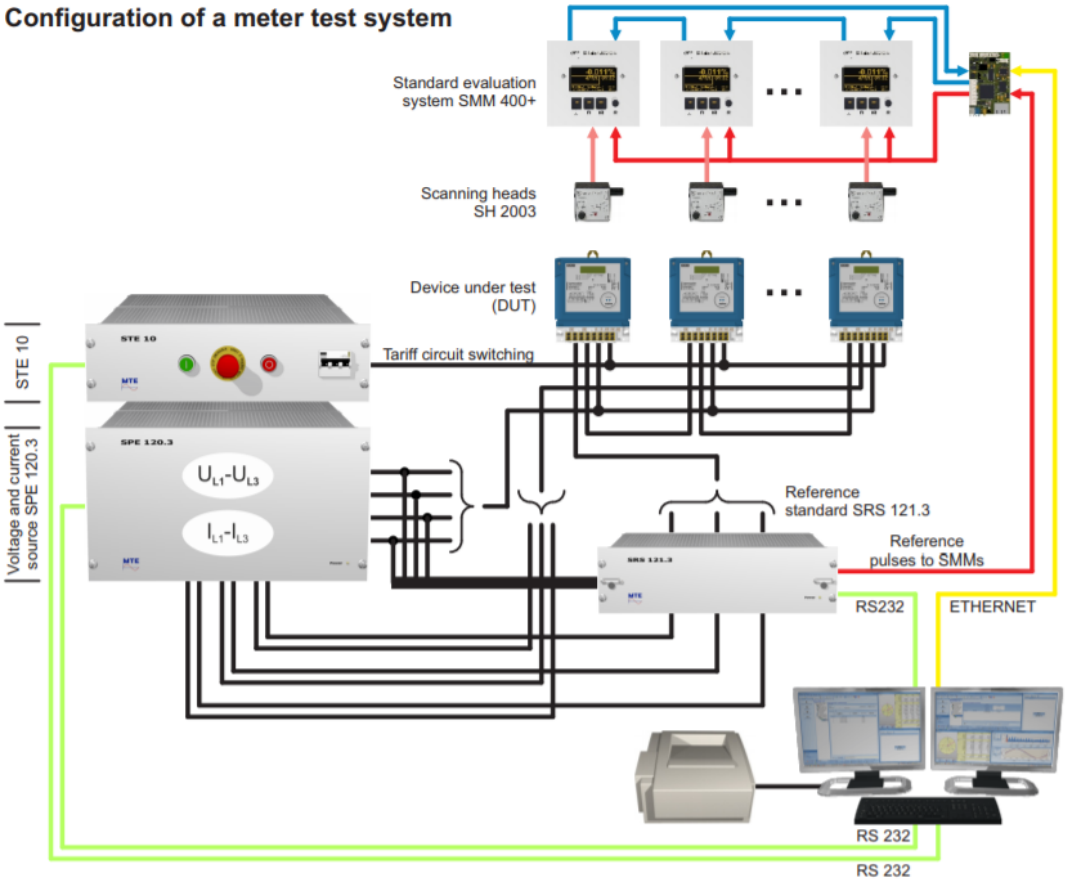


24 pav. Programos blokinė schema

2.4 Skaitiklio kalibravimo ir testavimo aparatūra

Atlikus elektros energijos matuoklio programavimą, jis buvo pratestuotas elektros energijos skaitiklių gamybos ir išmaniosios apskaitos sprendimų įmonėje „Elgama-Elektronika“, panaudojant stacionarų elektros energijos skaitiklių testavimo stendą, kurio galios šaltinis yra ypač tikslus MTE PSP-10, pavaizduotas 25 paveiksle, o 26 pav. pavaizduota testavimo sistemos konfigūracija.

25 pav. Stacionarus elektros energijos skaitiklių testavimo stendas [10]



26 pav. Stacionarios elektros energijos skaitiklių testavimo sistemos konfigūracija [10]

Elektros energijos matuoklių testavimo stendas MTE PSP-10 yra tikslus įtampos ir elektros srovės šaltinis, skirtas testuoti elektros skaitiklius. Tai vienfazis, kompiuteriu kontroliuojamas įtampos ir srovės šaltinis, kurio pagrindinės charakteristikos:

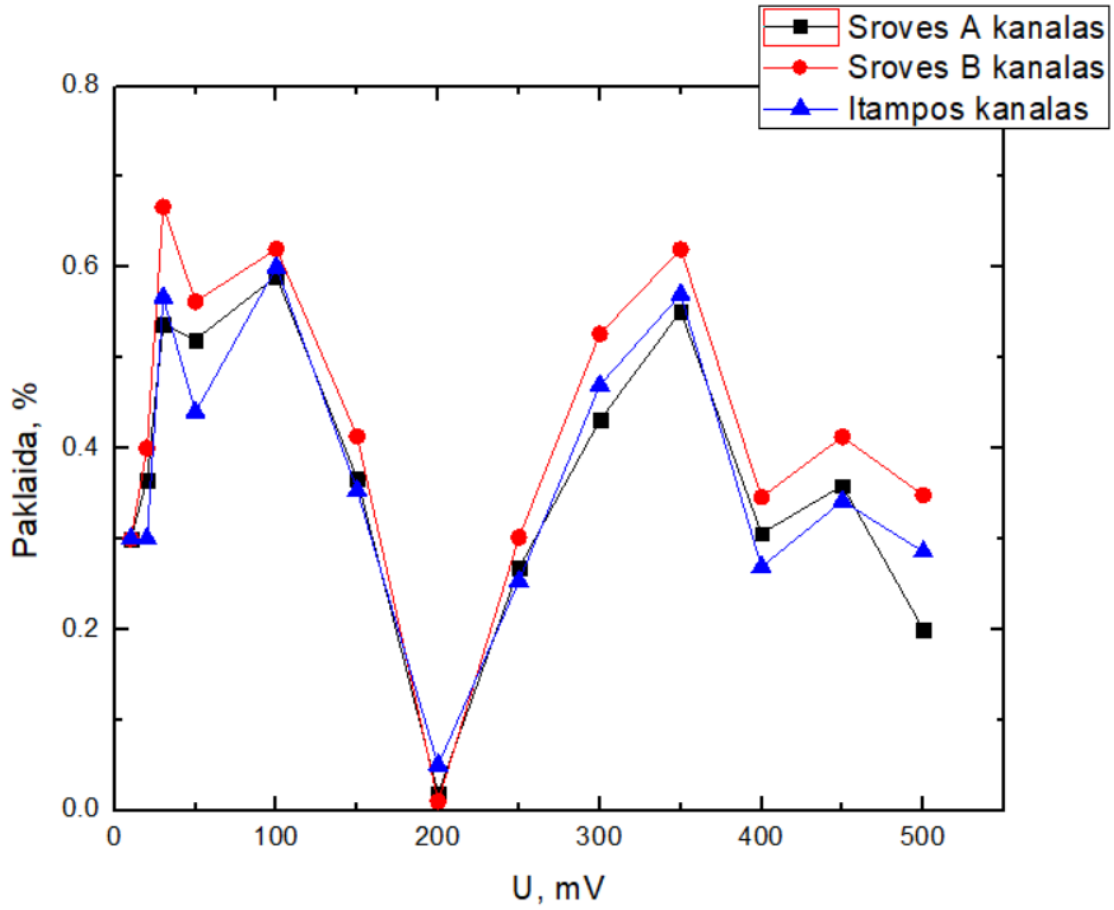
- Įtampa: 30 V iki 300 V;
- Srovė: 1 mA iki 100 A;
- Išėjimo galia:

  800 VA (įtampos)

  1200 VA (srovės)
- Galios efektyvumas > 85%.
- Šis galios šaltinis turi valymo bloką STE-10.
- Turi 1-3 galios šaltinius, su skaitmeniniais įtampos ir elektros srovės stiprintuvais. [10,11]

3. Darbo rezultatai ir jų aptarimas

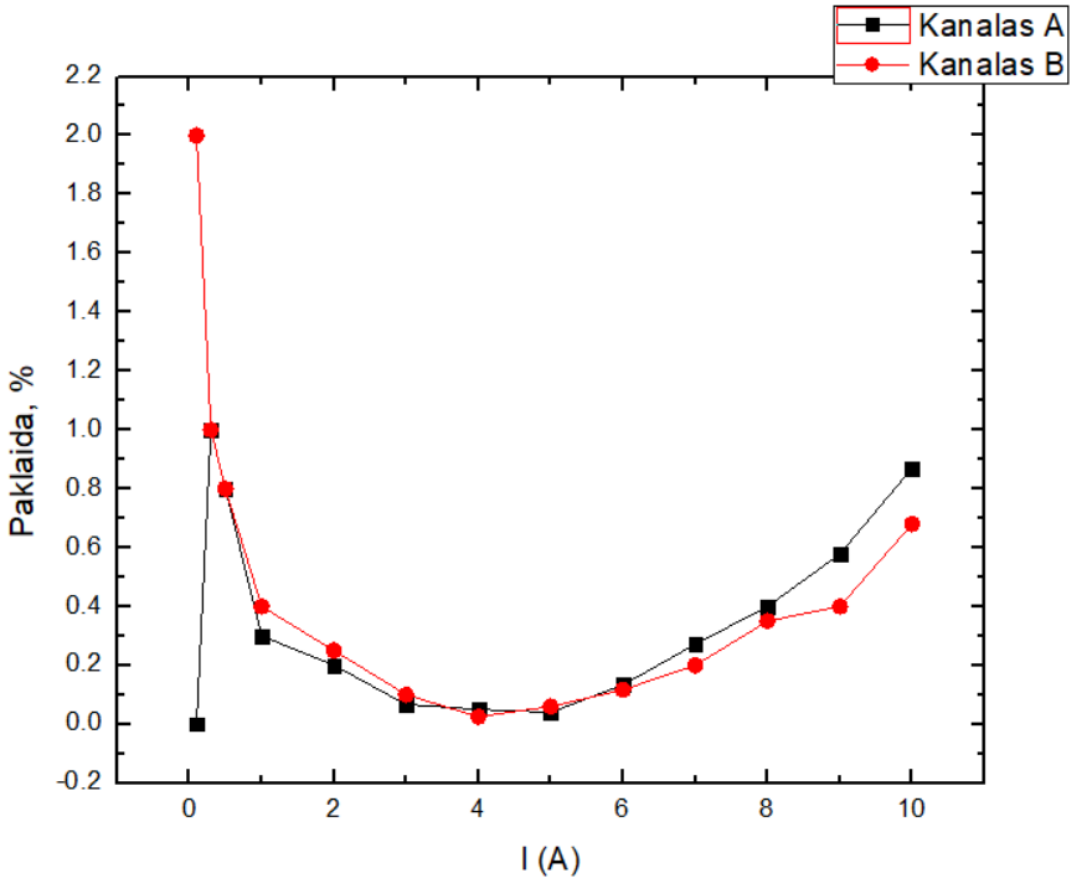
Sukonstravus bei suprogramavus elektros energijos matuoklį, buvo atlikti tyrimai, siekiant įvertinti pagaminto prietaiso veikimą bei tikslumą. Pirmiausia, prie trijų analoginių įėjimų (srovės A ir B kanalų, ir įtampos kanalo) buvo prijungtas funkcinis generatorius ir patikrintas surinkto skaitiklio veikimas visame darbiniame įtampos diapazone, nuo 0 mV iki 500 mV. Kalibravimas buvo atliktas 200 mV įtampos taške. 27 paveiksle parodytos skaitiklio fiksuotų signalų įtampos paklaidos priklausomybė nuo įtampos, paduotos iš funkcinio generatoriaus.



27 pav. Skaitiklio fiksuojamos įtampos paklaidos procentais priklausomybė nuo funkcinio generatoriaus generuojamos įtampos

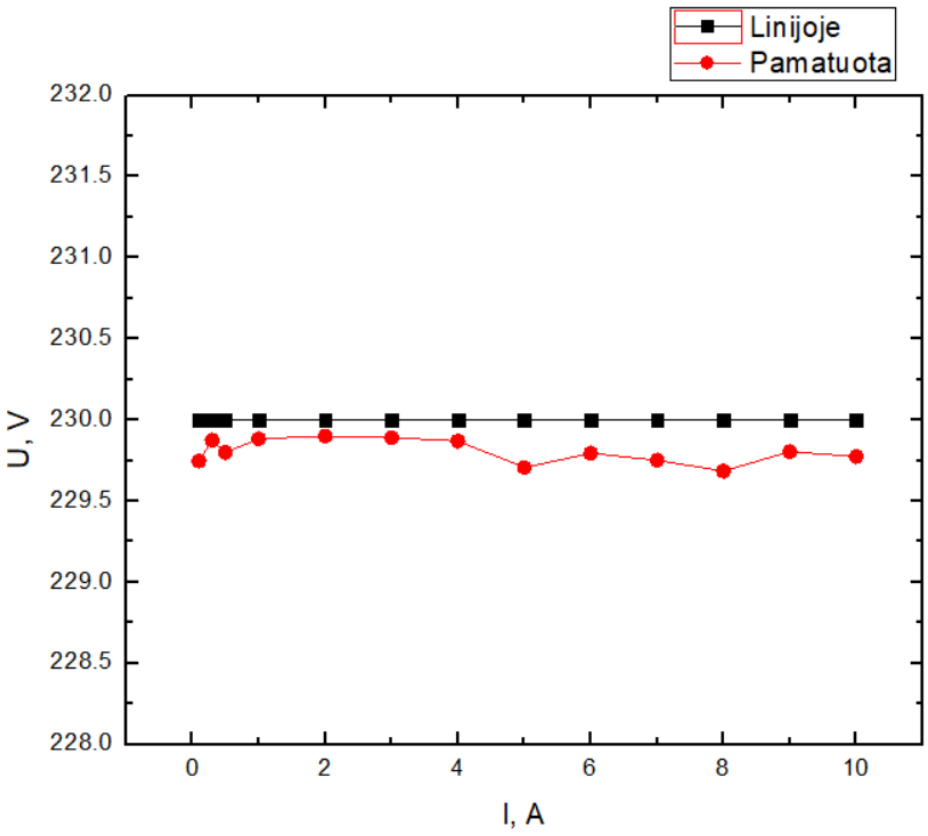
Šis bandymas buvo atliktas norint įsitikinti, kad schemoje naudojamas integrinis grandynas korektiškai dirba visame analoginių įėjimų įtampų ruože, o taip pat patikrinti komunikacijai tarp integrinio grandyno ADE7953 ir mikroprocesoriaus, atlikti pradinį testavimą. Iš 27 pav. matome, kad vidutinė paklaida srovės kanale A siekia 0,37 %, kanale B – 0,425 %, o įtampos kanale – 0,37 %. Gauti rezultatai parodo, kad sukonstruotas prietaisas korektiškai matuoja įeinantį signalą, tačiau visiškai pasitikėti gautais rezultatais negalime, kadangi naudotas funkcinis generatorius nėra visiškai tikslus įtampos šaltinis.

Kitame etape skaitiklio kalibravimas ir testavimas buvo atliekamas skaitiklį prijungus prie elektros energijos matuoklių kalibravimo stendo, elektros energijos skaitiklių gamybos ir išmaniosios apskaitos sprendimų įmonėje „Elgama-Elektronika“. Šio stendo pagalba galime tiksliai nustatyti elektros srovę, įtampą bei galią linijoje. Prietaisas buvo sukalibruotas 5 A srovės taške. Pirmojo bandymo metu buvo įvertinti abiejų srovės kanalų ir įtampos matavimų tikslumai, srovę linijoje keičiant nuo 0,1 A iki 10 A, o įtampą nustačius 230 V, esant aktyvinei apkrovai. Bandymo rezultatai pateikti 28, 29 ir 30 paveiksluose.

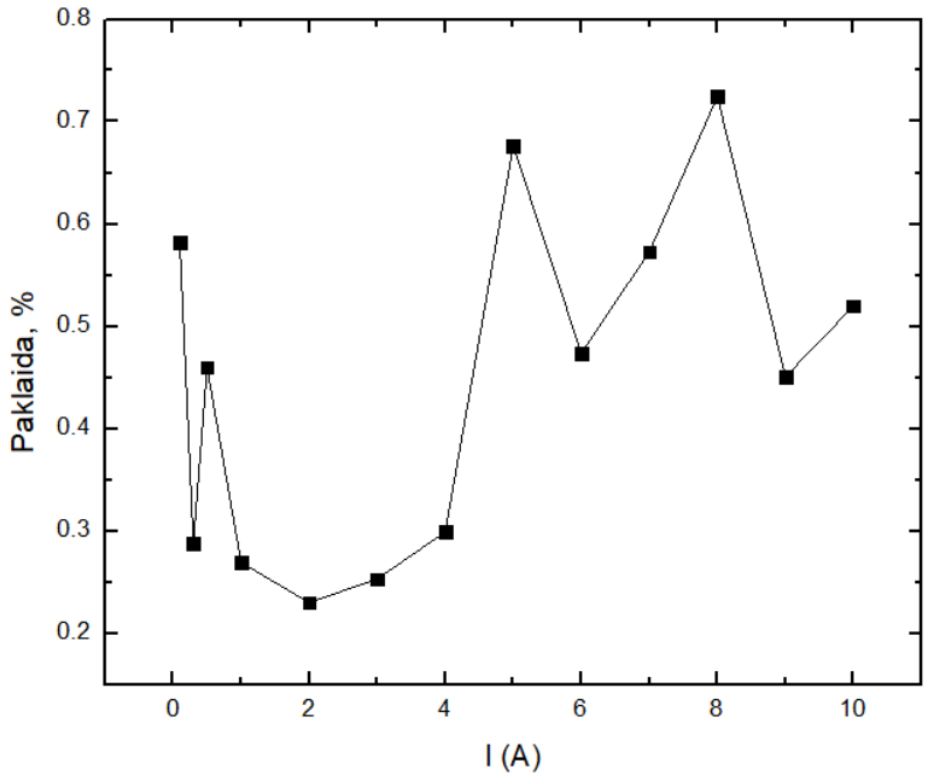


28 pav. Pamatuotos srovės paklaidos procentais priklausomybė nuo srovės linijoje

Iš 28 paveikslo matome, kad srovei linijoje esant nuo 1 A iki 10 A, matavimo paklaida abiejuose srovės kanaluose yra <1%, tačiau paklaida, srovei esant 0,1 A šokteli iki 2% kanale B. Tai gali būti sąlygota srovės sensoriaus Talesma ASM-010, kadangi gamintojo deklaruotas srovės korektiško veikimo diapazonas yra nuo 1 A iki 10 A. Vidutinė paklaida visame matavimo ruože lygi 0,36 % srovės kanale A, ir 0,49 % srovės kanale B. Vidutinė paklaida, srovę keičiant nuo 1 A iki 10A siekia vos 0,29 % ir 0.26 %, A ir B srovės kanaluose atitinkamai.



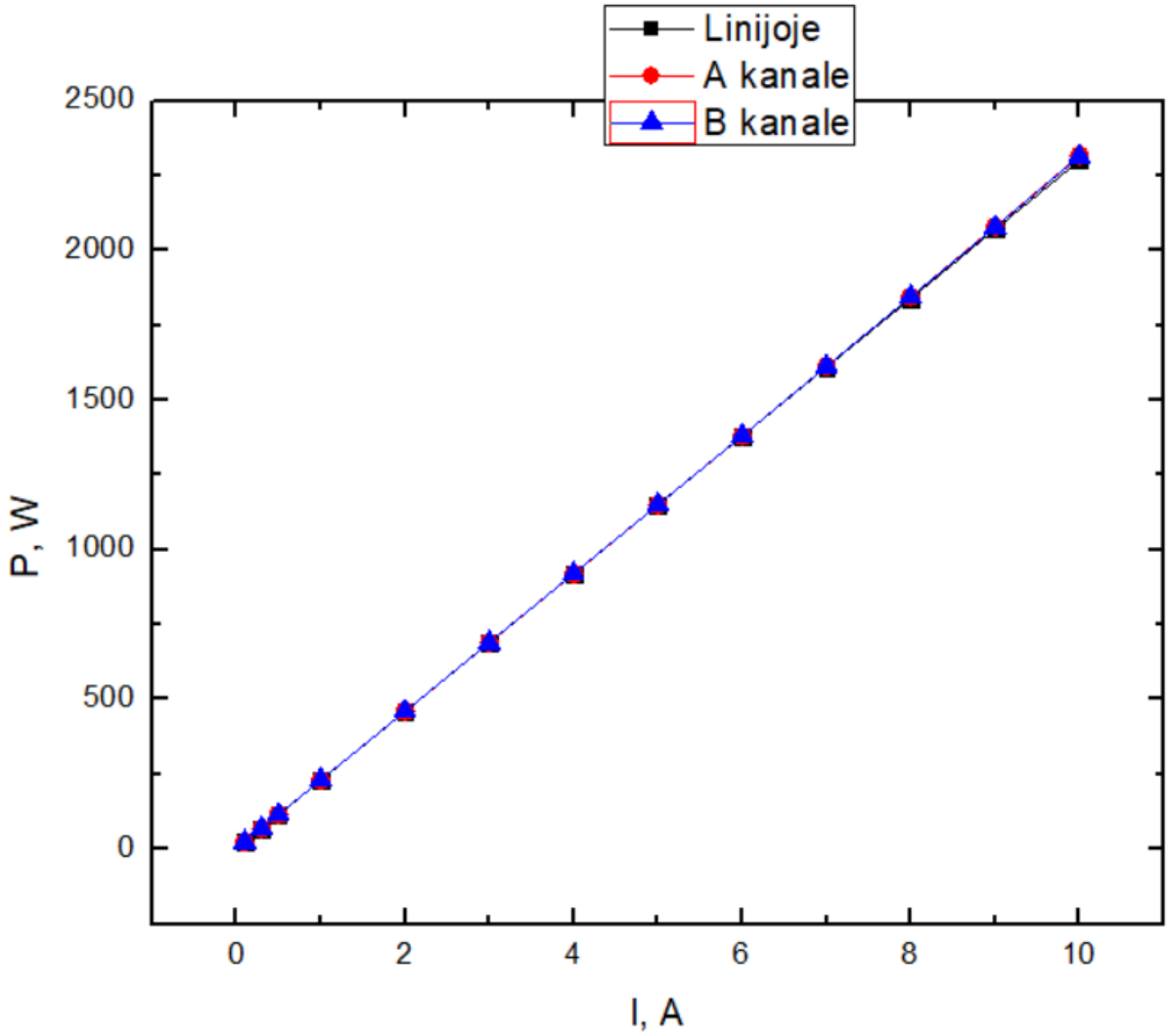
29 pav. Įtampos linijoje ir pamatuotos įtampos priklausomybė nuo srovės linijoje



30 pav. Pamatuotos įtampos paklaida procentais, keičiant srovę linijoje

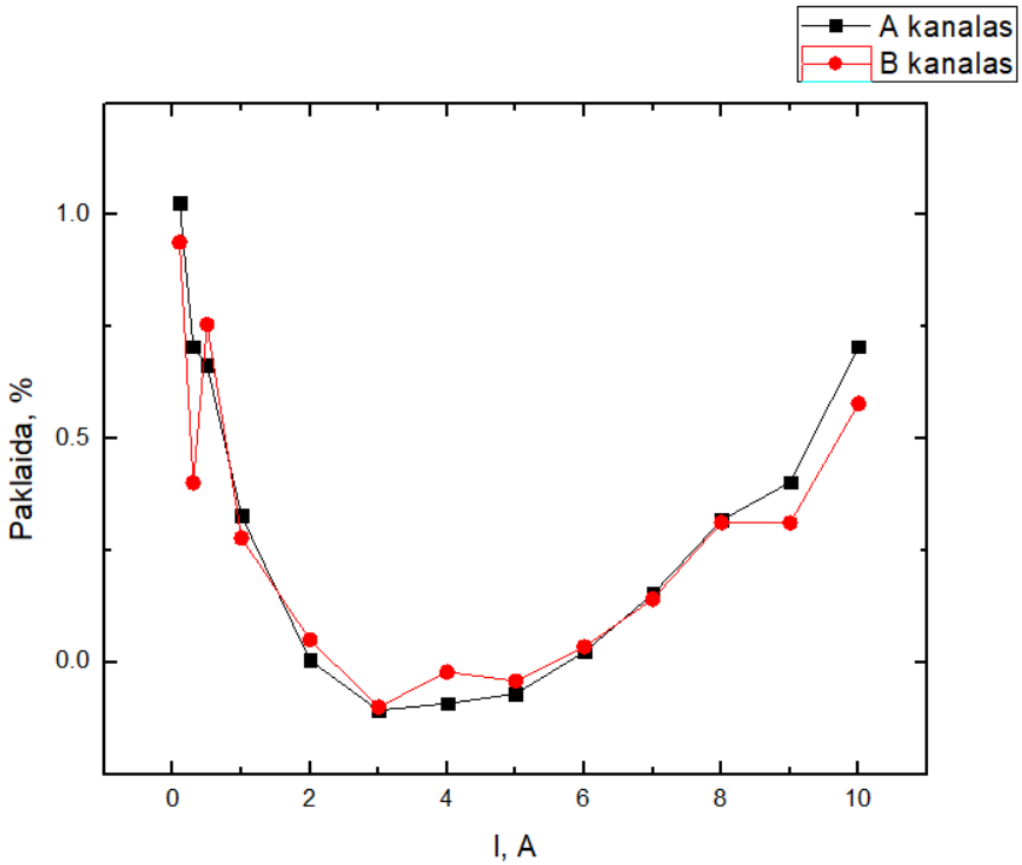
Iš 29 ir 30 paveikslų matome, kad srovės stiprio pokyčiai linijoje neturi didelės įtakos įtampos matavimų tikslumui, kadangi vidutinė įtampos matavimo paklaida visame matavimų ruože siekė 0,45%, o maksimali paklaida užfiksuota srovei esant 8 A – 0,72%.

Įvertinus srovės ir įtampos matavimų tikslumą, buvo matuojamos aktyvioji ir reaktyvioji galios, keičiant aktyvinės apkrovos vertę, t.y. srovę linijoje. Bandymo rezultatai pateikti 31, 32, 33 ir 34 pav.

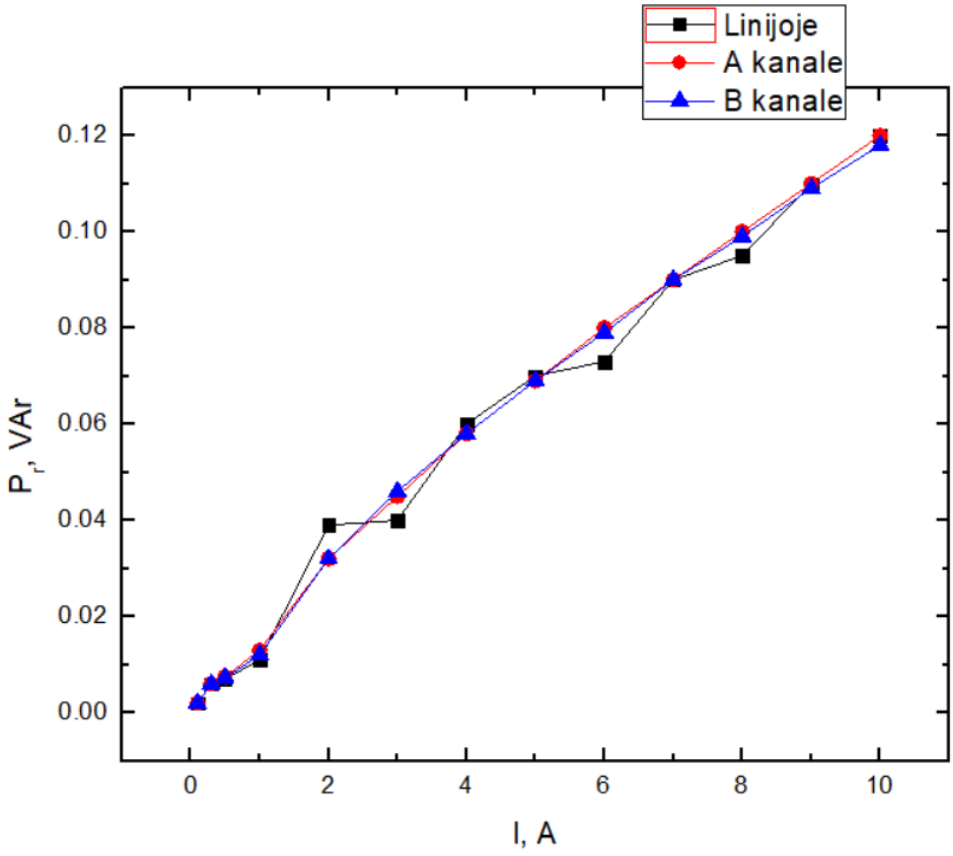


31 pav. Aktyviosios galios esančios linijoje ir pamatuotų galių priklausomybė nuo srovės stiprio

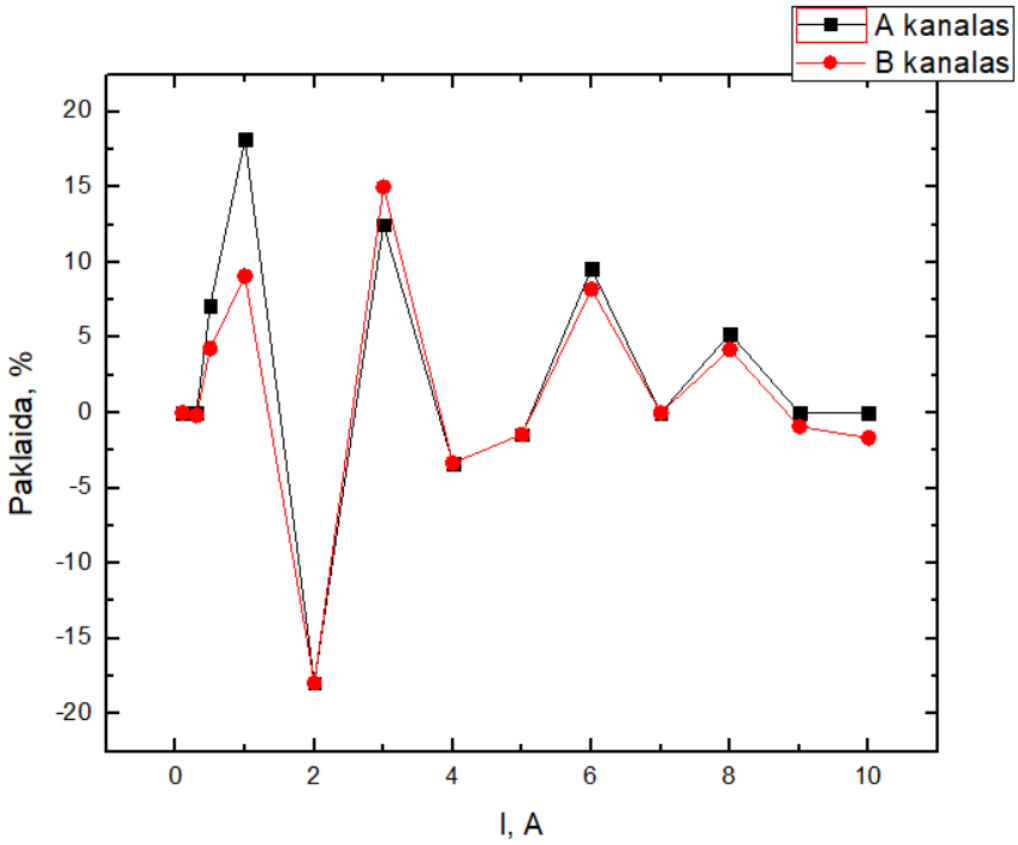
Iš 31 ir 32 paveikslų matome, kad aktyvinės apkrovos dydis neturi didelės įtakos matavimų tikslumui, kadangi visame matavimų ruože pamatuotos aktyviosios galios paklaida, neviršijo 1%, o vidutinė galios matavimų paklaida lygi 0,31 % srovės kanale A ir 0,28 % srovės kanale B.



32 pav. Aktyviosios galios paklaidos procentais priklausomybė nuo srovės stiprio linijoje



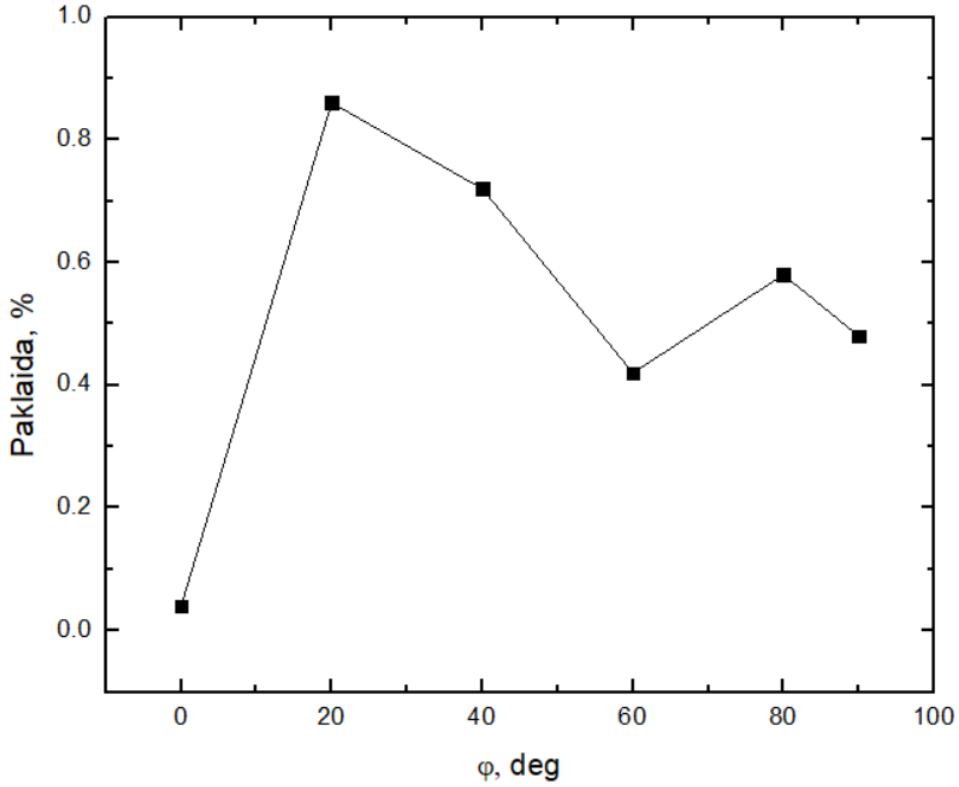
33 pav. Reaktyviosios galios esančios linijoje ir pamatuotų galių priklausomybė nuo srovės stiprio



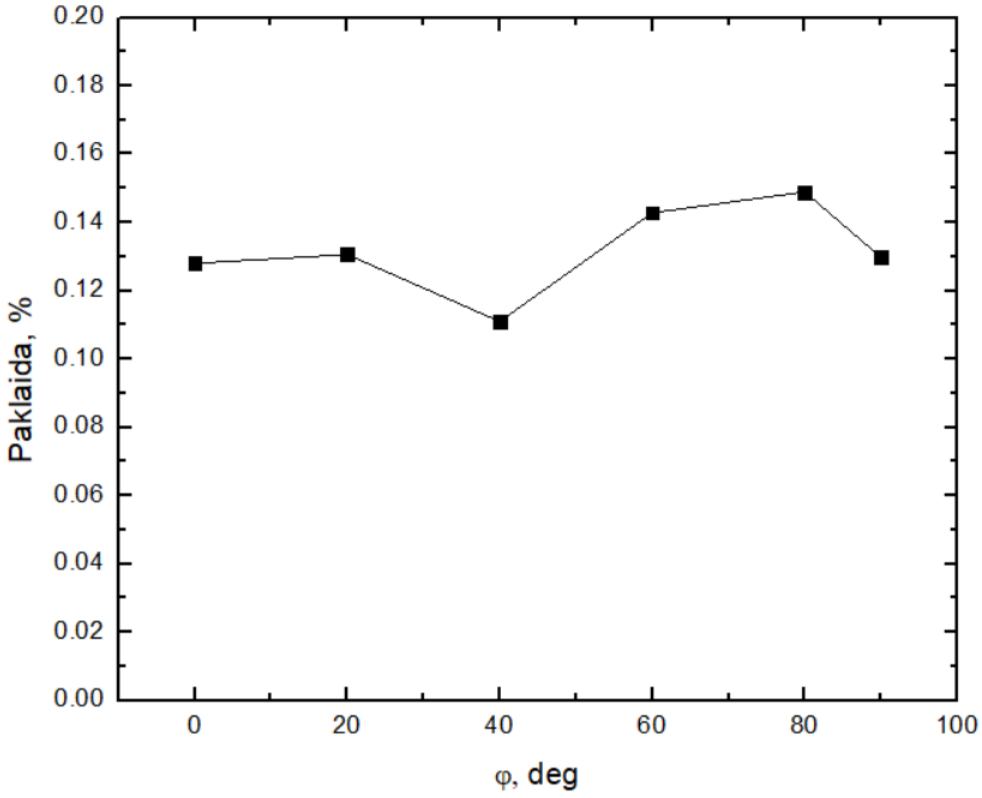
34 pav. Reaktyviosios galios paklaidos procentais priklausomybė nuo srovės stiprio linijoje

Reaktyviosios galios ir jos paklaidos procentais priklausomybė nuo srovės stiprio linijoje pavaizduoti 33 ir 34 paveiksluose atitinkamai. Matome, kad pamatuotos reaktyviosios galios vertės nuo realių verčių skiriasi vidutiniškai 5,8 % kanale A ir 5,09 % kanale B. Tokios didelės paklaidos atsiranda dėl to, kad buvo simuliuojama aktyvinė apkrova, tai yra fazių skirtumas tarp įtampos ir srovės yra lygus nuliui. Tokiu atveju visa energija, atėjusi į apkrovą yra pilnai sunaudojama, todėl reaktyviosios energijos vertės yra labai mažos, dėl to pamatuotoms reaktyviosios energijos vertėms didelę įtaką turi triukšmai sistemoje.

Išmatavus prietaiso tikslumą esant aktyviajai apkrovai, buvo pamatuotas skaitiklio tikslumas, egzistuojant fazės skirtumui tarp įtampos ir srovės, tai yra, simuliuojama apkrova nebėra tik aktyvinė. Šio matavimo metu buvo nustatyta 5 A elektros srovės stipris, 230 V įtampa. Tada buvo keičiamas fazių skirtumas tarp įtampos ir srovės, fiksuojamos skaitiklio matuojamos srovės ir įtampos vertės. Tuomet apskaičiuotos paklaidos tarp srovės ir įtampos, esančių linijoje verčių ir skaitiklio užfiksuotų verčių. Gauti rezultatai pavaizduoti 35 ir 36 paveiksluose:

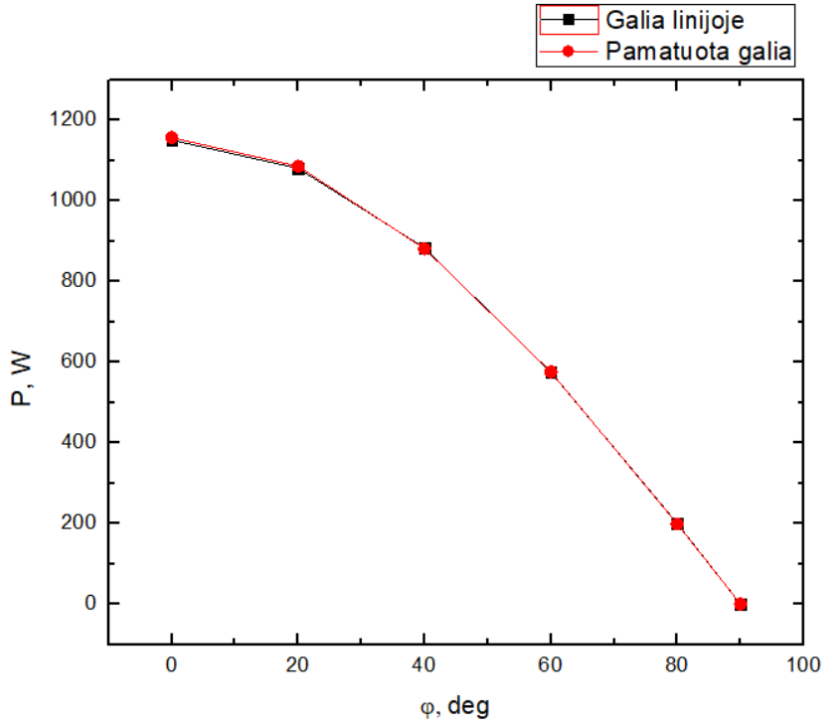


35 pav. Pamatuotos elektros srovės paklaidos priklausomybė nuo fazių skirtumo tarp srovės ir įtampos

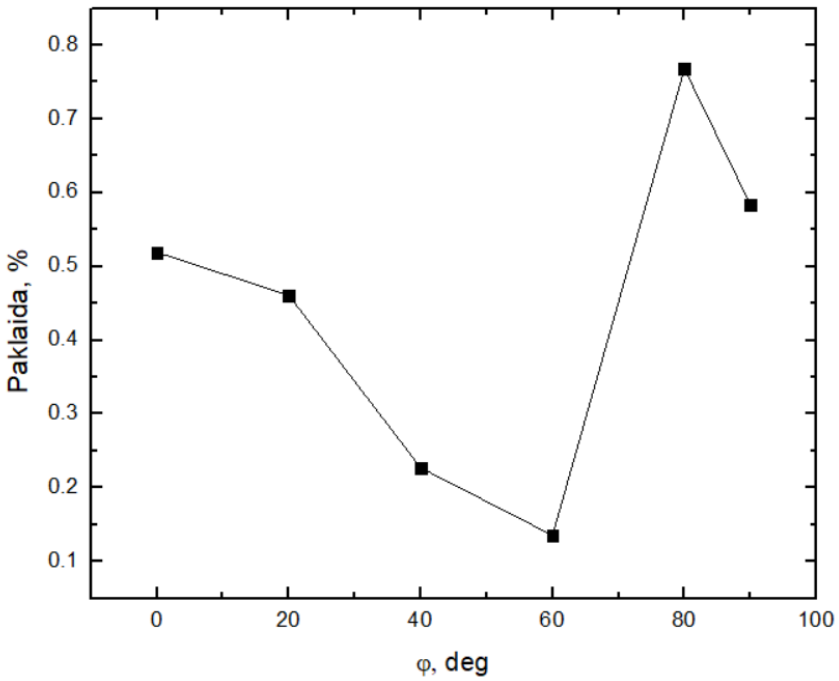


36 pav. Pamatuotos įtampos paklaidos priklausomybė nuo fazių skirtumo tarp srovės ir įtampos

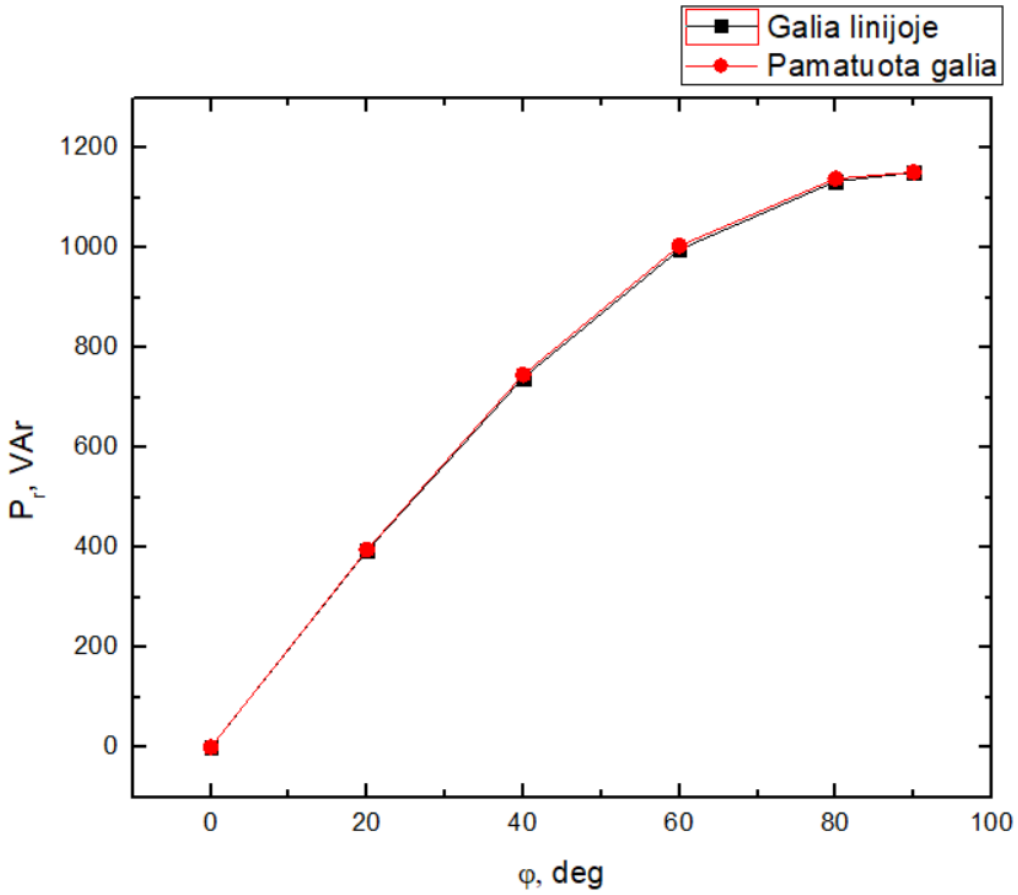
Iš 35, 36 paveikslų matome, kad išmatuotos elektros srovės paklaida yra <1%, o įtampos – svyruoja apie 0,14 %. Remdamiesi šiais duomenimis galime teigti, kad apkrovos tipas (aktyvioji ar reaktyvioji) neturi įtakos skaitiklio tikslumui, matuojant elektros srovę ir įtampą. Įsitikinus, kad srovę ir įtampą, esant įvairiems galios faktoriams, matuoklis matuoja korektiškai, buvo pamatuotos aktyviosios ir reaktyviosios galios vertės bei palygintos su šių galių vertėmis, esančiomis linijoje. Šių matavimų rezultatai pavaizduoti 37, 38, 39 ir 40 paveiksluose.



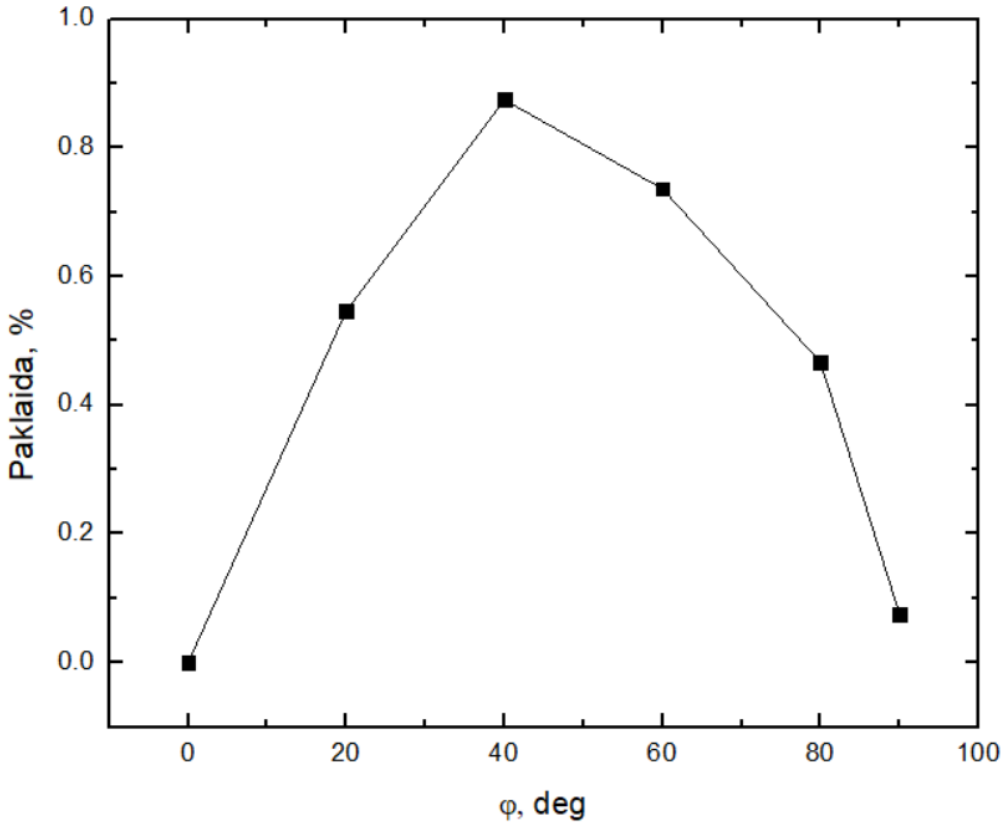
37 pav. Aktyviosios galios priklausomybė nuo fazių skirtumo tarp srovės ir įtampos



38 pav. Pamatuotos aktyviosios galios paklaidos priklausomybė nuo fazių skirtumo tarp srovės ir įtampos



39 pav. Reaktyviosios galios priklausomybė nuo fazių skirtumo tarp srovės ir įtampos

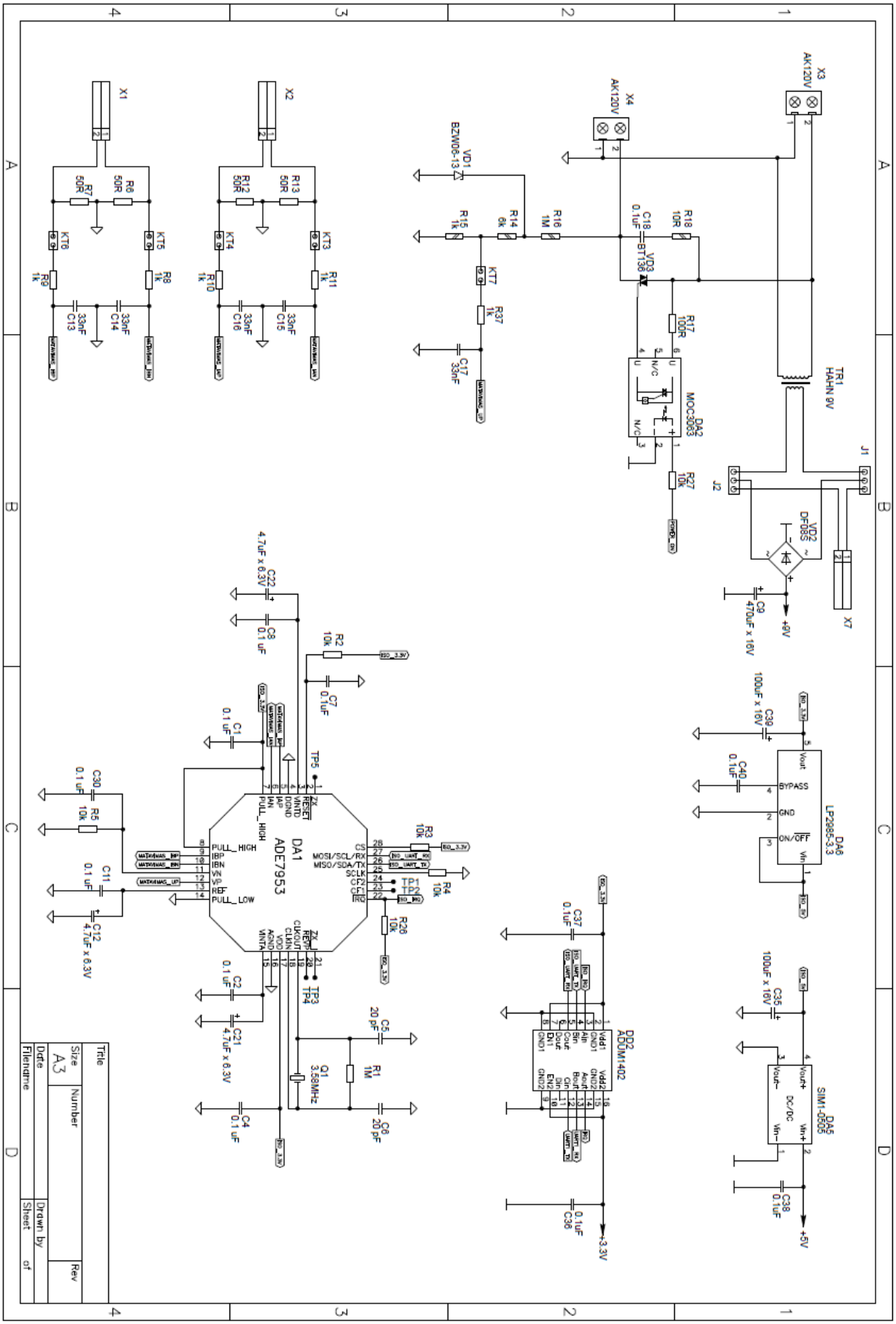


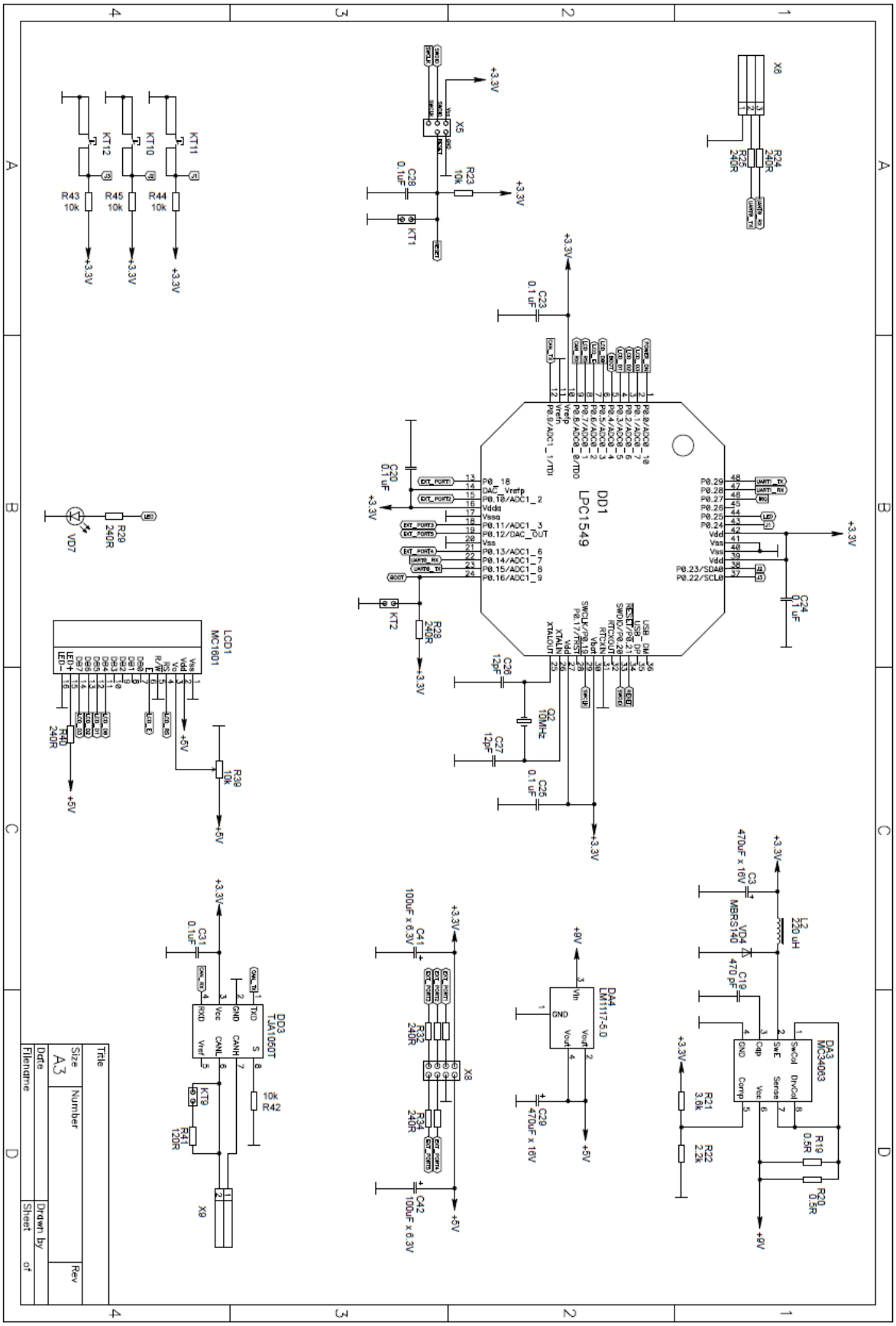
40 pav. Pamatuotos reaktyviosios galios paklaidos priklausomybė nuo fazių skirtumo tarp srovės ir įtampos

Iš 37 ir 38 paveikslų matome, kad aktyvioji galia yra matuojama leidžiamose tikslumo ribose, tai yra didžiausia matavimo paklaida užfiksuota kampui tarp srovės ir įtampos esant 80 laipsnių – 0,76 %. Iš 39 ir 40 paveikslų matome, kad skaitiklio užfiksuotos reaktyviosios galios vertės taip pat yra tikslios paklaidų ribose, didžiausia paklaida nustatyta esant 40 – ies laipsnių fazės skirtumui tarp srovės ir įtampos – 0,87%. Abiejų matavimų metu paklaidos neviršijo 1%.

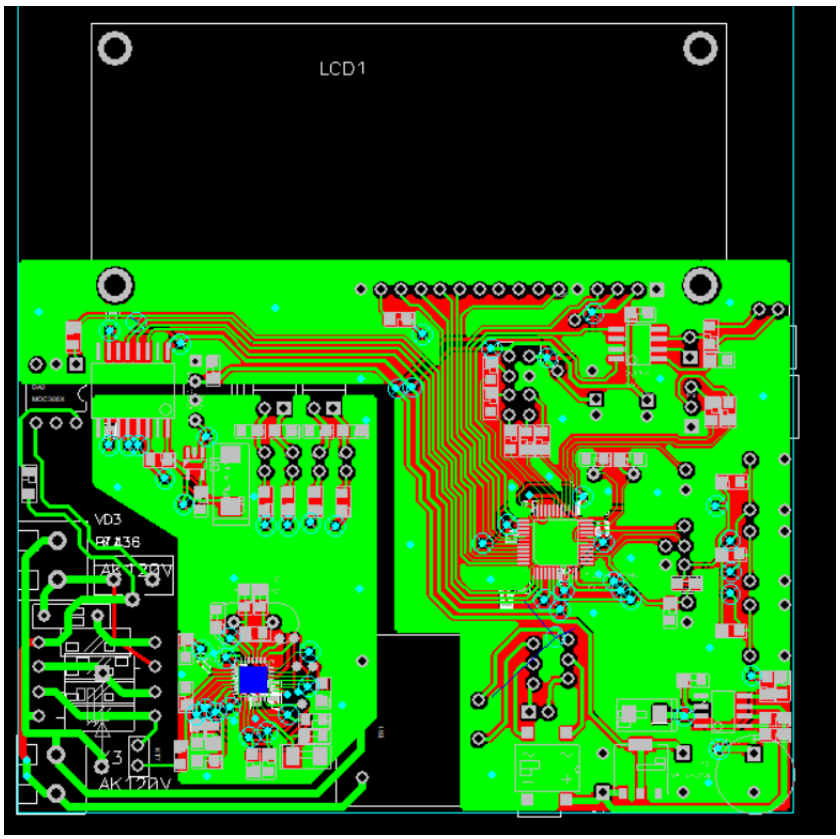
Iš šių rezultatų galime daryti išvadą, kad sukonstruoto skaitiklio tikslumo klasė yra 1.0. Tai reiškia, kad šio elektros energijos matuoklio paklaidos neviršija 1% visame matavimo diapazone, nepriklausomai nuo prijungtos apkrovos tipo (aktyvioji ar reaktyvioji).

3.1 Principinė schema ir PCB plokštė

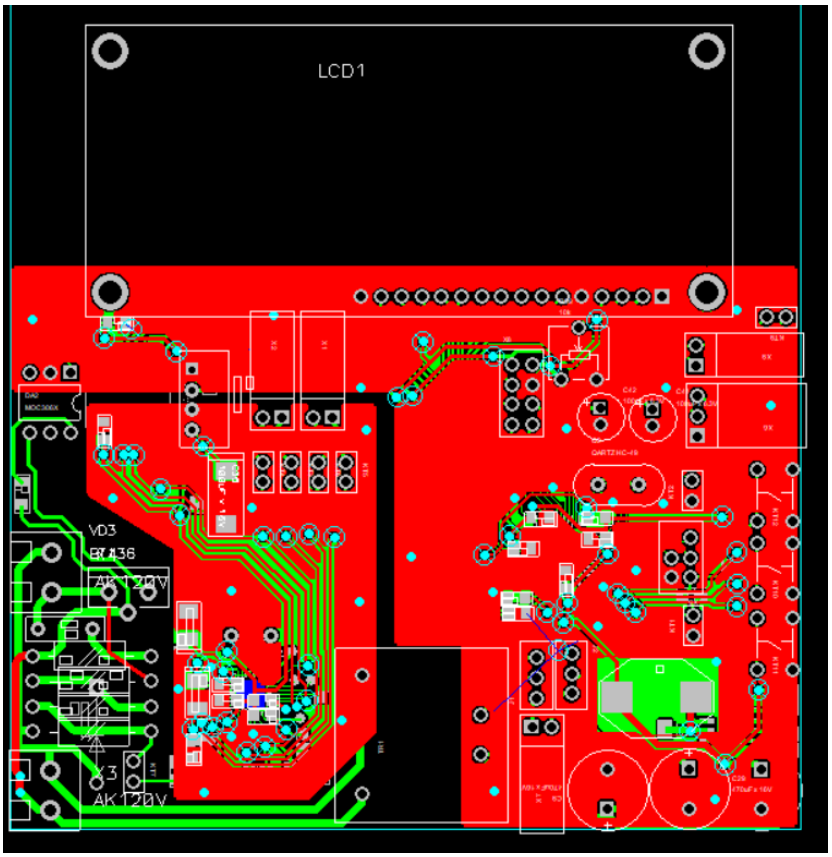




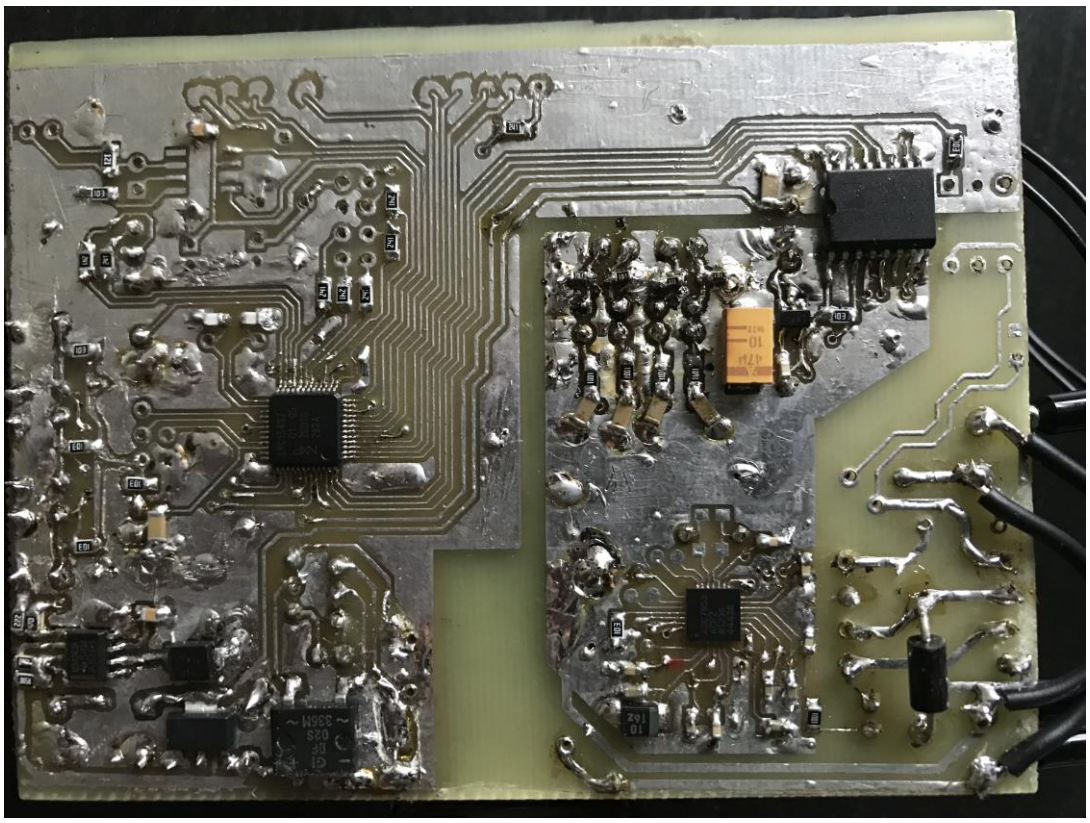
41 pav. elektros energijos skaitiklio principinė schema



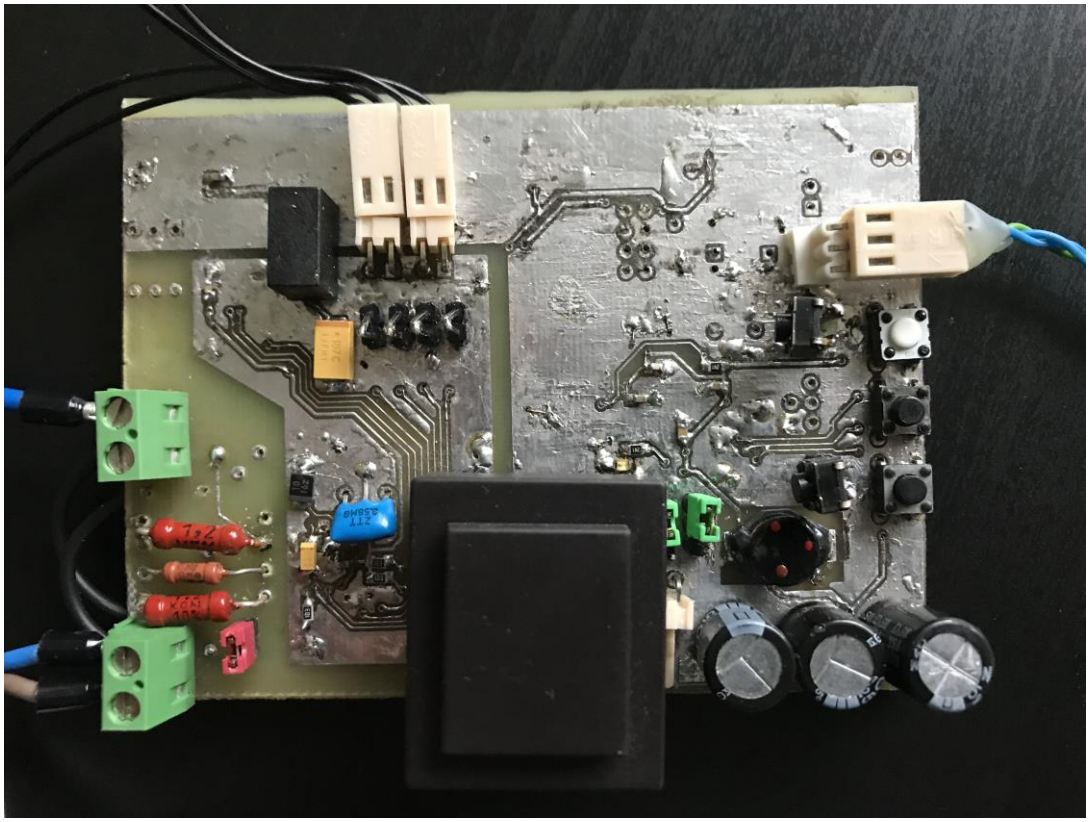
42 pav. PCB plokštės apatinis sluoksnis



43 pav. PCB plokštės viršutinis sluoksnis



44 pav. Sukonstruoto elektros energijos matuoklio apatinis sluoksnis



45 pav. Sukonstruoto elektros energijos matuoklio viršutinis sluoksnis

3. Išvados

1. Šio darbo metu buvo suprojektuota principinė elektros energijos matuoklio, skirto išmaniesiems namams, schema, PCB plokštė, sukonstruotas elektros energijos matuoklis.
2. Sukonstruoto matuoklio charakteristikos, esant aktyviajai apkrovai:
    - Srovės matavimo paklaida, ruože nuo 1A iki 10A – 0,87% kanale A ir 0,69% kanale B.
    - Įtampos matavimo paklaida – 0,72%.
    - Aktyviosios galios matavimo paklaida – 0.7 % kanale A ir 0.57% kanale B.
3. Sukonstruoto matuoklio charakteristikos, esant mišriai apkrovai:
    - Srovės matavimo paklaida – 0,86%.
    - Įtampos matavimo paklaida – 0,15%.
    - Aktyviosios galios matavimo paklaida – 0.76 %.
    - Reaktyviosios galios matavimo paklaida – 0.87 %
4. Sukonstruoto elektros energijos skaitiklio tikslumo klasė – 1.0.

4. Literatūros sąrašas

1. K.S.K. Weranga, S. Kumarawadu, D.P. Chandima, *Smart Metering Design and Applications*, Department of Electrical Engineering, University of Moratuwa, Moratuwa, Sri Lanka, 2013m, 141 psl.
2. Valentinas Zaveckas, *Elektrotechnikos pagrindai*, Vilnius, Vilniaus Gedimino Technikos Universitetas, 2012m, 90 psl.
3. M. Popa, H. Ciocarlie, A. S. Popa, and M. B. Racz, *Smart Metering for monitoring domestic utilities*, Timisoara, Romania, Politechnika Universitetas, 2010m.
4. F. Benzi, N. Anglani, E. Bassi, and L. Frosini, ―*Electricity Smart Meters Interfacing the Households*, IEEE Transactions on Industrial Electronics, vol. 58, no. 10, Oct. 2011m, pp. 4487–4494.
5. ADE7953 datasheet, Analog devices, 2016m.
6. K. Nagasarkar; M. S Suknija. 2005m. *Basic Electrical Engineering*. Oxford University Press. 634 p
7. Česlovas Pavasaris, *Puslaidininkiniai įtaisai. I dalis*, Vilnius, Vilniaus Universitetas, 2013m, 258 psl.
8. Boguslaw Pilich, *Engineering smart houses*, Lyngby, e Technical University of Denmark, 2004m, 62 psl.
9. Sergey y. Yurish, M.T.S.R. Gomez, *Smart sensors and MEMS*, NATO Scientific Affairs Division, 2005m., 418 psl.
10. MTE (Meter test equipment) stacionarių elektros energijos skaitiklių testavimo sistemų aprašymas. [Žiūrėta 2019 04 30] Prieiga: <https://www.mte.ch/data/files/ZPE%20Overview%20English_R03%20(09.2017).pdf>.
11. MTE PSP-10 galios šaltinio aprašymas. 2017m. [Žiūrėta 2019 04 30]Prieiga: <https://www.mte.ch/data/files/PSP%20System%20English_R02%20(10.2017).pdf>.
12. UART komunikacijos pagrindai. [Žiūrėta 2019 02 13]Prieiga: <http://www.circuitbasics.com/basics-uart-communication/>
13. Joe Campbell, *C Programmer's Guide to Serial Communications*, Michigan university, 1993m., 655 psl.
14. Išmaniųjų namų, automatikos apžvalga. [2018 05 20] Prieiga: <https://internetofthingsagenda.techtarget.com/definition/smart-home-or-building>
15. Česlovas Pavasaris, *Puslaidininkiniai įtaisai. 2 dalis*, Vilnius, Vilniaus Universitetas, 2009m, 453 psl.

Santrauka

Autorius: Edgardas Rinkevičius

Elektros energijos matuoklis, skirtas išmaniesiems namams.

Šio baigiamojo darbo tikslas – suprojektuoti, sukonstruoti ir pratestuoti elektros energijos matuoklį, skirtą išmaniesiems namams. Šiame darbe buvo apžvelgta išmaniųjų namų sąvoka, jų tipai. Taip pat elektros energijos matavimo principai, bei elektros energijos apskaitos būtinumas ir aktualumas. Darbe taip pat aptarti efektinės srovės bei įtampos, aktyviosios bei reaktyviosios galių ir energijų matavimo ir skaičiavimo principai. Šio darbo metu buvo suprojektuota elektros energijos matuoklio schema, pagaminta PCB plokštė ir sukonstruotas skaitiklis. PCB plokštė buvo gaminama fotorezistyviniu metodu. Sukonstruotas prietaisas realizuojamas aukšto tikslumo integrinio grandyno ADE7953.

Prietaisas buvo sukalibruotas ir pratestuotas elektros energijos skaitiklių gamybos ir išmaniosios apskaitos sprendimų įmonėje „Elgama-Elektronika“, į pagalbą pasitelkiant aukšto tikslumo galios šaltinį MTE PSP-10. Sukonstruotas elektros energijos matuoklis buvo pratestuotas esant aktyviajai apkrovai, o taip pat reaktyviajai apkrovai, keičiant fazių skirtumą tarp srovės ir įtampos nuo 0 iki 90 laipsnių.

Apibendrinant, darbo metu sukonstruoto elektros energijos skaitiklio tikslumas, esant prijungtai aktyviajai įtampai yra 0,87% kanale A ir 0,69% kanale B, matuojant elektros srovę, 0,72%, matuojant įtampą ir 0,7% ir 0.57% matuojant aktyviąją galią kanale A ir kanale B atitinkamai. Esant mišriai apkrovai, skaitiklio srovės matavimo paklaida – 0,86%, įtampos matavimo paklaida – 0,15%, aktyviosios galios – 0,76% ir reaktyviosios galios – 0.87%. Šios charakteristikos parodo, kad sukonstruoto elektros energijos skaitiklio tikslumo klasė yra 1.0.

Summary

Author: Edgardas Rinkevičius

Design, construction and testing of electric energy meter for smart houses.

The purpose of this thesis was to design an electric energy meter for smart houses, construct it and then test it. The thesis first examines the concept of smart houses, principles of electric energy metering as well as principles of root mean square measurements of voltage and electric current, measurements of active power, reactive power, active energy and reactive energy. PCB board was made using photoresist method. The electric energy meter designed is realized using high precision integrated circuit ADE7953, which is connected with microcontroller LPC1549.

The device was calibrated and tested in an electric energy metering company "Elgama-Electronics", using high accuracy power source MTE PSP-10. Electric energy meter's accuracy was tested using active load, as well as reactive load with phase difference between current and voltage ranging from 0 to 90 degrees.

To conclude, constructed electric energy meter measures current, when connected to active load, with an accuracy of 0,87 % in channel A and 0.69 % in channel B, voltage measurement accuracy is 0.72 %, and active power measurements 0.7 % and 0.57 % in channels A and B respectively. These characteristics put constructed energy meter in an accuracy class of 1.0.

Priedai

1 Priedas. Procesoriaus kojų sąrašas

```
#include "project_settings.h"
#include "project_include.h"

const MCU_PIN  DevicePins[]={
    { PIN_UART0_T X,  MOD_UART0,  SWM_UART0_TXD_O,  0},
    { PIN_UART0_RX,  MOD_UART0,  SWM_UART0_RXD_I,  PINMODE_PULLUP},
    { PIN_UART1_TX,  MOD_UART1,  SWM_UART1_TXD_O,  0},
    { PIN_UART1_RX,  MOD_UART1,  SWM_UART1_RXD_I,  PINMODE_PULLUP},
    {      PIN_LED,   MOD_GPIO,                0,  PINMODE_DIG_OUT_LOW},
    {            0,          0,                      0,  0}    // pabaigos
narys
};
```

2 Priedas. Procesoriaus kojų adresai, deklaruotos matavimo funkcijos

```
#define PIN_UART0_TX    0x00F
#define PIN_UART0_RX    0x00E
#define PIN_LED         0x019
#define PIN_UART1_TX    0x01D
#define PIN_UART1_RX    0x01C
#define PIN_CAN_TX      0x009
#define PIN_CAN_RX      0x008
#define PIN_LCD_D1      0x003
#define PIN_LCD_D2      0x002
#define PIN_LCD_D3      0x001
#define PIN_LCD_D0      0x005
#define PIN_LCD_E       0x006
#define PIN_LCD_RS      0x007
#define PIN_J1          0x018
#define PIN_J2          0x017
#define PIN_J3          0x016

extern const MCU_PIN  DevicePins[];

#include "ade7953.h"
int MeasCalValA (long RegValA);
int MeasCalValB (long RegValB);
int MeasCalValV (long RegValV);
int MeasCalValActEnA (long RegValActEnA);
int MeasCalValActEnB (long RegValActEnB);
```

```
int MeasCalValReActEnA (long RegValReActEnA);
int MeasCalValReActEnB (long RegValReActEnB);
int MeasCalValApEnA (long RegValApEnA);
int MeasCalValApEnB (long RegValApEnB);
int MeasCalValActPA (long RegValActPA);
int MeasCalValActPB (long RegValActPB);
int MeasCalValReActPA (long RegValReActPA);
int MeasCalValReActPB (long RegValReActPB);
int MeasCalValApPA (long RegValApPA);
int MeasCalValApPB (long RegValApPB);
```

3 Priedas. ADE7953 kojų sąrašas

```
#define ADE7953_SAGCYC              0x000
#define ADE7953_DISNOLOAD           0x001
#define ADE7953_LCYCMODE            0x004
#define ADE7953_PGA_V               0x007
#define ADE7953_PGA_IA              0x008
#define ADE7953_PGA_IB              0x009
#define ADE7953_WRITE_PROTECT       0x040
#define ADE7953_LAST_OP             0x0FD
#define ADE7953_LAST_RWDATA_8       0x0FF
#define ADE7953_Version             0x702
#define ADE7953_EX_REF              0x800

//16-bit Registers
#define ADE7953_ZXTOUT              0x100
#define ADE7953_LINECYC             0x101
#define ADE7953_CONFIG              0x102
#define ADE7953_CF1DEN              0x103
#define ADE7953_CF2DEN              0x104
#define ADE7953_CFMODE              0x107
#define ADE7953_PHCALA              0x108
#define ADE7953_PHCALB              0x109
#define ADE7953_PFA                 0x10A
#define ADE7953_PFB                 0x10B
#define ADE7953_ANGLE_A             0x10C
#define ADE7953_ANGLE_B             0x10D
#define ADE7953_Period              0x11E
#define ADE7953_ALT_OUTPUT          0x110
#define ADE7953_LAST_ADD            0x1FE
#define ADE7953_LAST_RWDATA_16      0x1FF
```

```
#define ADE7953_RESERVED           0x120

//*** CONFIG register
//*/ CFMODE register
//**/ALT_OUTPUT register

//24-bit and 32-bit registers
#define ADE7953_SAGLVL             0x200
#define ADE7953_ACCMODE            0x201
#define ADE7953_AP_NOLOAD          0x203
#define ADE7953_VAR_NOLOAD         0x204
#define ADE7953_VA_NOLOAD          0x205
#define ADE7953_AVA                0x210
#define ADE7953_BVA                0x211
#define ADE7953_AWATT              0x212
#define ADE7953_BWATT              0x213
#define ADE7953_AVAR               0x214
#define ADE7953_BVAR               0x215
#define ADE7953_IA                 0x216
#define ADE7953_IB                 0x217
#define ADE7953_V                  0x218
#define ADE7953_IRMSA              0x21A
#define ADE7953_IRMSB              0x21B
#define ADE7953_VRMS               0x21C
#define ADE7953_AENERGYA           0x21E
#define ADE7953_AENERGYB           0x21F
#define ADE7953_RENERGYA           0x220
#define ADE7953_RENERGYB           0x221
#define ADE7953_APENERGYA          0x222
#define ADE7953_APENERGYB          0x223
#define ADE7953_OVLVL              0x224
#define ADE7953_OILVL              0x225
#define ADE7953_VPEAK              0x226
#define ADE7953_RSTVPEAK           0x227
#define ADE7953_IAPEAK             0x228
#define ADE7953_RSTIAPEAK          0x229
#define ADE7953_IBPEAK             0x22A
#define ADE7953_RSTIBPEAK          0x22B
#define ADE7953_IRQENA             0x22C
#define ADE7953_IRQSTATA           0x22D
#define ADE7953_RSTIRQSTATA        0x22E
#define ADE7953_IRQENB             0x22F
#define ADE7953_IRQSTATB           0x230
```

```
#define ADE7953_RSTIRQSTATB         0x231
//#define ADE7953_CRC                 0x000
#define ADE7953_AIGAIN              0x280
#define ADE7953_AVGAIN              0x281
#define ADE7953_AWGAIN              0x282
#define ADE7953_AVARGAIN            0x283
#define ADE7953_AVAGAIN             0x284
#define ADE7953_AIRMSOS             0x286
#define ADE7953_VRMSOS              0x288
#define ADE7953_AWATTOS             0x289
#define ADE7953_AVAROS              0x28A
#define ADE7953_AVAOS               0x28B
#define ADE7953_BIGAIN              0x28C
#define ADE7953_BVGAIN              0x28D
#define ADE7953_BWGAIN              0x28E
#define ADE7953_BVARGAIN            0x28F
#define ADE7953_BVAGAIN             0x290
#define ADE7953_BIRMSOS             0x292
#define ADE7953_BWATTOS             0x295
#define ADE7953_BVAROS              0x296
#define ADE7953_BVAOS               0x297

extern volatile unsigned int  ADE7953_Timer;
void ADE7953_Write(int adr, int val);
int ADE7953_Read(int adr);
int Sign(int val);
int Sign2(int val);
```

4 Priedas. Pagrindinė main() programa

```
#include "project_settings.h"
#include "project_include.h"

volatile unsigned int MainTimer;
void Timer0Func(void);

int ADE7953_Read(int adr);
void ADE7953_Write(int adr, int val);
void Task_Uart(void);
void CalValA (long RegValA);
void CalValB (long RegValB);
void CalValV (long RegValV);
int main(void)
```

```
{
    short c;

    SYS_ClockInit();
    SYS_CalcCoreClock();
    PIN_Configure(MOD_GPIO);

    TIMER_Init(0,1000,Timer0Func);
    UART_Init(0, 9600);
    UART_Init(1, 4800);
    PeriodicalService_Init();
    CLI_Init();
    Hyperterm_Init(CLI_CHANEL, CLI_ProcessData);
    xfprint(OUTDEV_UART0,"\n\rLPC1549 Startas!!!");
    xfprint(OUTDEV_UART0,"\n\rclock = %d", SYS_GetCoreClock_Hz());

    ADE7953_Write(0xFE, 0xAD);
    ADE7953_Write(0x120, 0x30);
    ADE7953_Write(ADE7953_PHCALA, 0x83);
    ADE7953_Write(ADE7953_PHCALB, 0xA6);
    for(;;)
    {
        if(MainTimer>100)
        {
            MainTimer = 0;
            PIN_Toggle(PIN_LED);

        }
        //TIMER_Delay(1,1000);
        //PIN_Toggle(PIN_LED);
        //Task_Uart();
        Hyperterm_Scan();
    }
    return 0;
}
```

5 Priedas. Testavimo funkcija

```
void Task_Uart(void)
{
    short c;
    c = UART_Getch(0);
    if(c)
```

```
{
    c &= 0xFF;
    if(c == 'a')
    {
        xfprint(OUTDEV_UART0,"\n\radr=102 rd=%X", ADE7953_Read(0x102));
    }
    if(c == 'b')
    {
        xfprint(OUTDEV_UART0,"\n\radr=224 rd=%X", ADE7953_Read(0x224));
    }
    if(c == 'c')
    {
        xfprint(OUTDEV_UART0,"\n\radr=324 rd=%X", ADE7953_Read(0x324));
    }
    if(c == 'd')
    {
        xfprint(OUTDEV_UART0,"\n\rwrite 224");
        ADE7953_Write(0x224, 0x12345678);
    }
    if(c == 'e')
    {
        xfprint(OUTDEV_UART0,"\n\rwrite 324");
        ADE7953_Write(0x324, 0x12345678);
    }
    if(c == 'f')
    {
        xfprint(OUTDEV_UART0,"\n\raddr = %X", ADE7953_Read(0x1FE));
        xfprint(OUTDEV_UART0,"\n\rRW16 = %X", ADE7953_Read(0x1FF));
        xfprint(OUTDEV_UART0,"\n\rRW24 = %X", ADE7953_Read(0x2FF));
        xfprint(OUTDEV_UART0,"\n\rRW32 = %X", ADE7953_Read(0x3FF));
    }
    if(c == 'g')
    {
        xfprint(OUTDEV_UART0,"\n\rRMS voltage rd=%d", ADE7953_Read(0x21C));

    }
    if(c == 'h')
    {
        xfprint(OUTDEV_UART0,"\n\rRMS currentA rd=%d", ADE7953_Read(0x21A));
    }
    if(c == 'j')
    {
        xfprint(OUTDEV_UART0,"\n\rRMS currentB rd=%d", ADE7953_Read(0x21B));
```

```
        }
    }
}
```

6 Priedas. Rašymo ir nuskaitymo iš ADE7953 registrų funkcijos

```
#include "project_settings.h"
#include "project_include.h"

volatile unsigned int  ADE7953_Timer;
#define ADE7953_UART_DELAY  40
#define ADE7953_UART_NO     1
#define ADE7953_TIMER_NO    1

static void ADE7953_UartPutch(int val)
{
    switch(ADE7953_UART_NO)
    {
        #if USE_UART0>0
        case 0:
            UART_Putch(ADE7953_UART_NO, (char)val);
            while(UART0Status & UART_FULL_TX);
            break;
        #endif
        #if USE_UART1>0
        case 1:
            UART_Putch(ADE7953_UART_NO, (char)val);
            while(UART1Status & UART_FULL_TX);
            break;
        #endif
        #if USE_UART2>0
        case 2:
            UART_Putch(ADE7953_UART_NO, (char)val);
            while(UART2Status & UART_FULL_TX);
            break;
        #endif
        #if USE_UART3>0
        case 3:
            UART_Putch(ADE7953_UART_NO, (char)val);
            while(UART3Status & UART_FULL_TX);
            break;
        #endif
    }
}
```

```
int ADE7953_Read(int adr)
{
    int f1,f2,f3;
    ADE7953_UartPutch(0x35);
    TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
    ADE7953_UartPutch(adr >> 8);
    TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
    ADE7953_UartPutch(adr);

    ADE7953_Timer = 0;
    f1 = 0;
    f3 = 0;
    while(1)
    {
        f2 = UART_Getch(1);
        if(f2)
        {
            f2 &= 0xFF;
            f2 <<= 8 * f3;
            f1 |= f2;
            f3++;
        }
        if(ADE7953_Timer > 50) break;
    }
    return f1;
}
void ADE7953_Write(int adr, int val)
{
    int i,byte_count;
    ADE7953_UartPutch(0xCA);
    TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
    ADE7953_UartPutch(adr >> 8);
    TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
    ADE7953_UartPutch(adr);
    TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
    byte_count = (adr >> 8) & 3;
    for(i=0; i<byte_count+1; i++)
    {
        ADE7953_UartPutch(val);
        TIMER_Delay(ADE7953_TIMER_NO, ADE7953_UART_DELAY);
        val >>= 8;
    }
    Delay(10);
```

```
}
int Sign(int val)
{
    if(val & 0x800000) val |= 0xFF000000;
    return val;
}
int Sign2(int val)
{
    if(val & 0x8000) val |= 0xFFFF0000;
    return val;
}
```

7 Priedas. Matavimo ir kalibravimo funkcijos

```
int CalIA = 433065;
void CalValA (long RegValA)
{
    CalIA = RegValA;
}
int CalIB = 429000;
void CalValB (long RegValB)
{
    CalIB = RegValB;

}
int CalV = 5908200;
void CalValV (long RegValV)
{
    CalV = RegValV;

}
int CalActEnA = 0;
int CalActEnB = 0;
int CalApEnA = 0;
int CalApEnB = 0;
int CalReActEnA = 0;
int CalReActEnB = 0;
int CalActPA = 152505;
int CalActPB = 150960;
int CalReActPA = -5858;
int CalReActPB = -6915;
int CalApPA = 152838;
int CalApPB = 151200;
```

```
int MeasCalValA (long RegValA)
    {
        long long V;
        long long U;
        V = 5000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValA * V / (long long)CalIA;
        return (int)U;
    }
    int MeasCalValB (long RegValB)
    {
        long long V;
        long long U;
        V = 5000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValB * V / (long long)CalIB;
        return (int)U;
    }
    int MeasCalValV (long RegValV)
    {
        long long V;
        long long U;
        V = 230000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValV * V / (long long)CalV;
        return (int)U;
    }

    int MeasCalValActEnA (long RegValActEnA)
    {
        long long V;
        long long U;
        V = 1875000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValActEnA * V / (long long)CalActEnA;
        return (int)U;
    }

        int MeasCalValActEnB (long RegValActEnB)
    {
        long long V;
        long long U;
        V = 1875000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValActEnB * V / (long long)CalActEnB;
        return (int)U;
    }
    int MeasCalValReActEnA (long RegValReActEnA)
```

```
{
    long long V;
    long long U;
    V = 1875000; //įtampos amplitudė kalibravimo taške
    U = (long long)RegValReActEnA * V / (long long)CalReActEnA;
    return (int)U;
}
int MeasCalValReActEnB (long RegValReActEnB)
{
    long long V;
    long long U;
    V = 1875000; //įtampos amplitudė kalibravimo taške
    U = (long long)RegValReActEnB * V / (long long)CalReActEnB;
    return (int)U;
}
    int MeasCalValApEnA (long RegValApEnA)
{
    long long V;
    long long U;
    V = 1875000; //įtampos amplitudė kalibravimo taške
    U = (long long)RegValApEnA * V / (long long)CalApEnA;
    return (int)U;
}
    int MeasCalValApEnB (long RegValApEnB)
{
    long long V;
    long long U;
    V = 1875000; //įtampos amplitudė kalibravimo taške
    U = (long long)RegValApEnB * V / (long long)CalApEnB;
    return (int)U;
}
    int MeasCalValActPA (long RegValActPA)
{
    long long V;
    long long U;
    V = 1150000; //mW
    U = (long long)RegValActPA * V / (long long)CalActPA;
    return (int)U;
}
    int MeasCalValActPB (long RegValActPB)
{
    long long V;
    long long U;
```

```
        V = 1150000; //mW amplitudė kalibravimo taške
        U = (long long)RegValActPB * V / (long long)CalActPB;
        return (int)U;
    }
    int MeasCalValReActPA (long RegValReActPA)
    {
        long long V;
        long long U;
        V = 70; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValReActPA * V / (long long)CalReActPA;
        return (int)U;
    }
        int MeasCalValReActPB (long RegValReActPB)
    {
        long long V;
        long long U;
        V = 70; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValReActPB * V / (long long)CalReActPB;
        return (int)U;
    }
        int MeasCalValApPA (long RegValApPA)
    {
        long long V;
        long long U;
        V = 1150000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValApPA * V / (long long)CalApPA;
        return (int)U;
    }
        int MeasCalValApPB (long RegValApPB)
    {
        long long V;
        long long U;
        V = 1150000; //įtampos amplitudė kalibravimo taške
        U = (long long)RegValApPB * V / (long long)CalApPB;
        return (int)U;
    }
```

8 Priedas. Valdymo per komandinę eilutę kodas.

```
#include "project_settings.h"
#include "project_include.h"
long EnergijaA;
long EnergijaB;
int CalValA (long RegValA);
```

```
int CalValB (long RegValB);
int CalValV (long RegValV);
// Process command
void CLI_ProcessCommand_Ext(CLI_DATA *data)
{
    int val1, val2, flag = 0;
    if(CLI_StrCmp("get",data,0))
    {
        if(CLI_StrCmp("config",data,1))
        {
            xfprintnl(CLI_CHANEL,"COFIG = %08d",  ADE7953_Read(ADE7953_CONFIG));
            flag = 1;
        }
        if(CLI_StrCmp("reserved",data,1))
        {
            xfprintnl(CLI_CHANEL,"reserved = %08d",
ADE7953_Read(ADE7953_RESERVED));
            flag = 1;
        }
        if(CLI_StrCmp("lastadd",data,1))
        {
            xfprintnl(CLI_CHANEL,"lastadd = %08d",
ADE7953_Read(ADE7953_LAST_ADD));
            flag = 1;
        }
        if(CLI_StrCmp("cf1den",data,1))
        {
            xfprintnl(CLI_CHANEL, "CF1DEN = %08d",
ADE7953_Read(ADE7953_CF1DEN));
            flag = 1;
        }
        if(CLI_StrCmp("IA",data,1))
        {
            xfprintnl(CLI_CHANEL, "IA = %08d", ADE7953_Read(ADE7953_IA));
            flag = 1;
        }
        if(CLI_StrCmp("IB",data,1))
        {
            xfprintnl(CLI_CHANEL, "IB = %08d", ADE7953_Read(ADE7953_IB));
            flag = 1;
        }
        if(CLI_StrCmp("V",data,1))
        {
```

```
            xfprintnl(CLI_CHANEL, "V = %08d", ADE7953_Read(ADE7953_V));
            flag = 1;
        }
        if(CLI_StrCmp("IRMSA",data,1))
        {
            xfprintnl(CLI_CHANEL, "IRMSA = %08d", ADE7953_Read(ADE7953_IRMSA));
            flag = 1;
        }
        if(CLI_StrCmp("IRMSB",data,1))
        {
            xfprintnl(CLI_CHANEL, "IRMSB = %08d", ADE7953_Read(ADE7953_IRMSB));
            flag = 1;
        }
        if(CLI_StrCmp("VRMS",data,1))
        {
            xfprintnl(CLI_CHANEL, "VRMS = %08d", ADE7953_Read(ADE7953_VRMS));
            flag = 1;
        }
        if(CLI_StrCmp("AENERGYA",data,1))
        {
            val1 = Sign(ADE7953_Read(ADE7953_AENERGYA));
            EnergijaA += val1;
            xfprintnl(CLI_CHANEL, "AENERGYA = %08d", val1);
            flag = 1;
        }
        if(CLI_StrCmp("AENERGYB",data,1))
        {
            val1 = Sign(ADE7953_Read(ADE7953_AENERGYB));
            EnergijaB += val1;
            xfprintnl(CLI_CHANEL, "AENERGYB = %08d",
Sign(ADE7953_Read(ADE7953_AENERGYB)));
            flag = 1;
        }
        if(CLI_StrCmp("RENERGYA",data,1))
        {
            xfprintnl(CLI_CHANEL, "RENERGYA = %08d",
Sign(ADE7953_Read(ADE7953_RENERGYA)));
            flag = 1;
        }
        if(CLI_StrCmp("RENERGYB",data,1))
        {
            xfprintnl(CLI_CHANEL, "RENERGYB = %08d",
Sign(ADE7953_Read(ADE7953_RENERGYB)));
```

```
            flag = 1;
        }
        if(CLI_StrCmp("APENERGYA",data,1))
        {
            xfprintnl(CLI_CHANEL, "APENERGYA = %08d",
Sign(ADE7953_Read(ADE7953_APENERGYA)));
            flag = 1;
        }
        if(CLI_StrCmp("APENERGYB",data,1))
        {
            xfprintnl(CLI_CHANEL, "APENERGYB = %08d",
Sign(ADE7953_Read(ADE7953_APENERGYB)));
            flag = 1;
        }
        if(CLI_StrCmp("AVA",data,1))
        {
            xfprintnl(CLI_CHANEL, "AVA = %08d", ADE7953_Read(ADE7953_AVA));
            flag = 1;
        }
        if(CLI_StrCmp("BVA",data,1))
        {
            xfprintnl(CLI_CHANEL, "BVA = %08d", ADE7953_Read(ADE7953_BVA));
            flag = 1;
        }
        if(CLI_StrCmp("AWATT",data,1))
        {
            xfprintnl(CLI_CHANEL, "AWATT = %d",
Sign(ADE7953_Read(ADE7953_AWATT)));
            flag = 1;
        }
        if(CLI_StrCmp("BWATT",data,1))
        {
            xfprintnl(CLI_CHANEL, "BWATT = %08d",
Sign(ADE7953_Read(ADE7953_BWATT)));
            flag = 1;
        }
        if(CLI_StrCmp("AVAR",data,1))
        {
            xfprintnl(CLI_CHANEL, "AReactP = %08d",
Sign(ADE7953_Read(ADE7953_AVAR)));
            flag = 1;
        }
        if(CLI_StrCmp("BVAR",data,1))
        {
```

```
            xfprintnl(CLI_CHANEL, "BReactP = %08d",
Sign(ADE7953_Read(ADE7953_BVAR)));
            flag = 1;
        }
        if(CLI_StrCmp("ANGLE_A",data,1))
        {
            xfprintnl(CLI_CHANEL, "ANGLE_A = %08d",
Sign2(ADE7953_Read(ADE7953_ANGLE_A)));
            flag = 1;
        }
        if(CLI_StrCmp("ANGLE_B",data,1))
        {
            xfprintnl(CLI_CHANEL, "ANGLE_B = %08d",
Sign2(ADE7953_Read(ADE7953_ANGLE_B)));
            flag = 1;
        }
        if(CLI_StrCmp("EnergijaA",data,1))
        {
            xfprintnl(CLI_CHANEL, "EnergijaA = %08d", EnergijaA);
            flag = 1;
        }
        if(CLI_StrCmp("EnergijaB",data,1))
        {
            xfprintnl(CLI_CHANEL, "EnergijaB = %08d", EnergijaB);
            flag = 1;
        }
        if(CLI_StrCmp("PVA",data,1))
        {
            xfprintnl(CLI_CHANEL, "PowFactorA = %08d",
Sign(ADE7953_Read(ADE7953_PFA)));
            flag = 1;
        }
        if(CLI_StrCmp("PVB",data,1))
        {
            xfprintnl(CLI_CHANEL, "PowFactorB = %08d",
Sign(ADE7953_Read(ADE7953_PFB)));
            flag = 1;
        }
    }
    if(CLI_StrCmp("set",data,0))
    {
        if(CLI_StrCmp("config",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
```

```
            ADE7953_Write(ADE7953_CONFIG, val1);
            xfprintnl(CLI_CHANEL,"set CONFIG = %08X", val1);
            flag = 1;
        }
        if(CLI_StrCmp("AIGAIN",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
            ADE7953_Write(ADE7953_AIGAIN, val1);
            xfprintnl(CLI_CHANEL,"set AIGAIN = %08X", val1);
            flag = 1;
        }
        if(CLI_StrCmp("AVGAIN",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
            ADE7953_Write(ADE7953_AVGAIN, val1);
            xfprintnl(CLI_CHANEL,"set AVGAIN = %08X", val1);
            flag = 1;
        }
        if(CLI_StrCmp("BIGAIN",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
            ADE7953_Write(ADE7953_BIGAIN, val1);
            xfprintnl(CLI_CHANEL,"set BIGAIN = %08X", val1);
            flag = 1;
        }
        if(CLI_StrCmp("PHCALA",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
            ADE7953_Write(ADE7953_PHCALA, val1);
            xfprintnl(CLI_CHANEL,"set PHCALA = %08d", val1);
            flag = 1;
        }
        if(CLI_StrCmp("PHCALB",data,1))
        {
            val1 = CLI_ScanHex(data, 2);
            ADE7953_Write(ADE7953_PHCALB, val1);
            xfprintnl(CLI_CHANEL,"set PHCALB = %08d", val1);
            flag = 1;
        }
    }
    if(CLI_StrCmp("cal",data,0))
    {
        if(CLI_StrCmp("IA",data,1))
```

```
        {
            xfprintnl(CLI_CHANEL,"Srove A sukalibruota = %08d",
CalValA(ADE7953_Read(ADE7953_IRMSA)));
            flag = 1;
        }
        if(CLI_StrCmp("IB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Srove B sukalibruota = %08d",
CalValB(ADE7953_Read(ADE7953_IRMSB)));
            flag = 1;
        }
        if(CLI_StrCmp("V",data,1))
        {
            xfprintnl(CLI_CHANEL,"Srove V sukalibruota = %08d",
CalValV(ADE7953_Read(ADE7953_VRMS)));
            flag = 1;
        }
    }
    if(CLI_StrCmp("meas",data,0))
    {
        if(CLI_StrCmp("IA",data,1))
        {
            xfprintnl(CLI_CHANEL,"Srove A = %08d",
MeasCalValA(ADE7953_Read(ADE7953_IRMSA)));
            flag = 1;
        }
        if(CLI_StrCmp("IB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Srove B = %08d",
MeasCalValB(ADE7953_Read(ADE7953_IRMSB)));
            flag = 1;
        }
        if(CLI_StrCmp("V",data,1))
        {
            xfprintnl(CLI_CHANEL,"Itampa V = %08d",
MeasCalValV(ADE7953_Read(ADE7953_VRMS)));
            flag = 1;
        }
        if(CLI_StrCmp("AENA",data,1))
        {
            xfprintnl(CLI_CHANEL,"Akt. En. A = %08d",
MeasCalValActEnA(ADE7953_Read(ADE7953_AENERGYA)));
            flag = 1;
        }
```

```
        if(CLI_StrCmp("AENB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Akt. En. B = %08d",
MeasCalValActEnB(ADE7953_Read(ADE7953_AENERGYB)));
            flag = 1;
        }
        if(CLI_StrCmp("APENA",data,1))
        {
            xfprintnl(CLI_CHANEL,"Piln. En. A = %08d",
MeasCalValApEnA(ADE7953_Read(ADE7953_APENERGYA)));
            flag = 1;
        }
        if(CLI_StrCmp("APENB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Piln. En. B = %08d",
MeasCalValApEnB(ADE7953_Read(ADE7953_APENERGYB)));
            flag = 1;
        }
        if(CLI_StrCmp("RENA",data,1))
        {
            xfprintnl(CLI_CHANEL,"ReAkt. En. A = %08d",
MeasCalValReActEnA(ADE7953_Read(ADE7953_RENERGYA)));
            flag = 1;
        }
        if(CLI_StrCmp("RENB",data,1))
        {
            xfprintnl(CLI_CHANEL,"ReAkt. En. B = %08d",
MeasCalValReActEnB(ADE7953_Read(ADE7953_RENERGYB)));
            flag = 1;
        }
        if(CLI_StrCmp("APA",data,1))
        {
            xfprintnl(CLI_CHANEL,"Akt. Galia A = %08d",
MeasCalValActPA(ADE7953_Read(ADE7953_AWATT)));
            flag = 1;
        }
        if(CLI_StrCmp("APB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Akt. Galia B = %08d",
MeasCalValActPB(ADE7953_Read(ADE7953_BWATT)));
            flag = 1;
        }
        if(CLI_StrCmp("APPA",data,1))
        {
```

```
            xfprintnl(CLI_CHANEL,"Piln. Galia A = %08d",
MeasCalValApPA(ADE7953_Read(ADE7953_AVA)));
            flag = 1;
        }
        if(CLI_StrCmp("APPB",data,1))
        {
            xfprintnl(CLI_CHANEL,"Piln. Galia B = %08d",
MeasCalValApPB(ADE7953_Read(ADE7953_BVA)));
            flag = 1;
        }
        if(CLI_StrCmp("RPA",data,1))
        {
            xfprintnl(CLI_CHANEL,"ReAkt. Galia A = %08d",
MeasCalValReActPA(Sign(ADE7953_Read(ADE7953_AVAR))));
            flag = 1;
        }
        if(CLI_StrCmp("RPB",data,1))
        {
            xfprintnl(CLI_CHANEL,"ReAkt. Galia B = %08d",
MeasCalValReActPB(Sign(ADE7953_Read(ADE7953_BVAR))));
            flag = 1;
        }
    }
    if(flag == 0) xfprintnl(CLI_CHANEL, "unknown command");
    CLI_ShowWelcome(); }
```

9 Priedas. Nustatymų failas

```
#include "mcu_list.h"

#define CHIP_TYPE           CHIP_LPC1549
#define STACK_SIZE          200
#define HEAP_SIZE           200
#define RTC_FREQUENCY_HZ    32768
#define OSC_FREQUENCY_MHZ   10
#define OSC_FREQUENCY_HZ    (OSC_FREQUENCY_MHZ*1000000)
/*
#define OSCFG_MAIN_CLOCK_SETUP 1
#define OSCFG_USB_CLOCK_SETUP  0
#define OSCFG_SCT_CLOCK_SETUP  0

#define OSCCFG_RANGE          0
#define OSCCFG_BYPASS         0
// OSCCFG_PLL_CLK_SEL: 0-IRC, 1-crystal
```

```
#define OSCCFG_PLL_CLK_SEL    0
// OSCFG_MAIN_CLK_SEL_A: 0-IRC, 1-system, 2-WD
#define OSCFG_MAIN_CLK_SEL_A  1
#define OSCFG_MAIN_CLK_SEL_B  2
#define OSCCFG_MAIN_PSEL      1
#define OSCCFG_MAIN_MSEL      6
#define OSCCFG_MAIN_CLK_DIV   1*/


#define OSCFG_MAIN_CLOCK_SETUP  1
#define OSCFG_USB_CLOCK_SETUP   0
#define OSCFG_SCT_CLOCK_SETUP   0

#define OSCCFG_RANGE            0
#define OSCCFG_BYPASS           0
// OSCCFG_PLL_CLK_SEL: 0-IRC, 1-crystal
#define OSCCFG_PLL_CLK_SEL      0
// OSCFG_MAIN_CLK_SEL_A: 0-IRC, 1-system, 2-WD
#define OSCFG_MAIN_CLK_SEL_A    0
#define OSCFG_MAIN_CLK_SEL_B    2
#define OSCCFG_MAIN_PSEL        1
#define OSCCFG_MAIN_MSEL        5
#define OSCCFG_MAIN_CLK_DIV     1

#define USE_UART0           1
#define RX0BUFSIZE          100
#define TX0BUFSIZE          1000

#define USE_UART1           1
#define RX1BUFSIZE          100
#define TX1BUFSIZE          200

#define USE_TIMER0          1
#define USE_TIMER1          1
#define USE_DELAY           1
#define USE_OS              0
#define USE_STREAM          1
#define USE_ERROR           1
#define ERROR_UART          0
#define ERROR_BITRATE       9600
#define ERROR_TIMER         0

#define USE_PERIODICAL_SERVICE 1
```

```
#define USE_HYPERTERMINAL   1
#define USE_CLI             1
#define CLI_CHANEL          OUTDEV_UART0
#define CLI_BUF_SIZE        128
```