Përmbajtja

Abstrakti 1
Abstract 2
Programet e përdorura 3
Pajisjet e përdorura 3
1. Hyrje 4
2. Përpunimi analog i tensionit dhe rrymës 6
2.1 Amplifikatori operacional TL072 6
2.2 Matja e tensionit - Stadi hyrës 6
2.3 Matja e tensionit – Përpunimi analog dhe shndërrimi i brezit 8
2.4 Matja e rrymës – Stadi hyrës 9
2.5 Matja e rrymës – Përpunimi analog dhe shndërrimi i brezit 10
3. Përpunimi digjital i tensionit dhe rrymës 11
3.1 Mikrokontrolleri AT89C51 11
3.2 Protokoli i komunikimit SPI 12
3.3 Shndërruesi A/D MCP3208 15
4. Arkitektura softuerike 17
4.1 Modulet dhe funksionet 17
4.2 Algoritmet themelore 20
4.3 Llogaritja e fuqisë dhe energjisë 22
5. Shfaqja e vlerave 24
5.1 LCD LM044L 24
6. Ruajta e të dhënave 26
6.1 Protokoli i komunikimit I2C 26
6.2 EEPROM FM24C256 28
Skema 30
7. Qarqet e integruara 31
Përfundim 33
Referencat 34
Shtojca A: Leximi i përmbajtjes së EEPROM 35
Shtojca B: Kodi për AT89C51 37

Abstrakti

Në punim prezantohet zhvillimi i një pajisje mikrokompjuterike për matjen, shfaqjen dhe regjistrimin e tensionit, rrymës, fuqisë dhe energjisë për rrymë DC, me qëllim demonstrimin e parimit të matjes së energjisë për pajisje të ngjashme, aplikimin e qarqeve analoge bazike, integrimin qarqeve për shndërrimin dhe përpunim e sinjaleve duke përdorur mikrokontrollerin e familjes 8051, pjesët tjera bazike të përdorura në implementim përfshijnë sensorë të tensionit dhe rrymës, si dhe një ndërfaqe për shfaqjen e të dhënave në një ekran. Edhe pse pajisja është projektuar për përdorim eksperimental dhe në kushte të kontrolluara, ajo mund të shërbejë si një pikënisje për zhvillimin e pajisjeve më të avancuara për matjen dhe menaxhimin e energjisë në sisteme të automatizuara. Duke u bazuar në këtë konfiguracion të dizajnuar dhe rezultatet e nxjerra, pajisja mund të ofrojë saktësi dhe besueshmëri për implementime të modifikuara mbi të.

Abstract

This paper presents the development of a microcomputer-based device for measuring, displaying, and recording voltage, current, power, and energy for DC power sources, with the aim of demonstrating the principle of energy measurement for similar devices, applying basic analog circuits, and integrating circuits for signal conversion and processing using an 8051 family microcontroller. Other basic components used in the implementation include voltage and current sensors, as well as an interface for displaying data on a screen. Although the device was designed for experimental use in controlled conditions, it can serve as a starting point for the development of more advanced devices for energy measurement and management in automated systems. Based on this designed configuration and the results obtained, the device can offer accuracy and reliability for modified implementations.

Programet e përdorura

Proteus v8.11

SDCC

Pajisjet e përdorura

TL072

MCP3208

AT89C51

LM044L

FM24C256

1. Hyrje

Matja e energjisë elektrike është një element që luan rol të rëndësishëm në monitorimin dhe menaxhimin e konsumit të energjisë në sisteme të ndryshme. Përveç aplikimeve të zakonshme në industrinë e energjisë, matësit e energjisë janë të nevojshëm edhe për përdorime në pajisje me qëllim të optimizimit të konsumit dhe efikasitetit. Megjithatë, ndonëse ka pasur përparime të dukshme në teknologjitë e matjes, ka mjaft sfida që karakterizojnë dizjanin, sidomos në lidhje me sigurinë e matjes, saktësinë dhe mundësitë për monitorimin në kohë reale te të dhënave.

Një shembull tipik i pajisjëve për matjen e energjisë është iEM3250, është pajisje kompakte me saktësi të lartë për matjen e energjisë dhe monitorimin në kohë reale të konsumit të energjisë. Mat në mënyrë të vazhdueshme tensionin, rrymën, fuqinë (aktive dhe reaktive) dhe frekuencën e sinjalit. Shfaq energjinë në kWh dhe kVAh. Mund të komunikoj sipas protokolit të komunikimin Modbus RTU dhe Modbus TCP, poashtu mund të monitorohet dhe kontrollohet nga distanca.

Fig. 1 – Matësi i energjisë iEM3250 [12]

Rezultatet shfaqen në një ekran LCD. Ndërsa pajisja mund të kontrollohet lokalisht përmes butonave. Pajisje të tjera të ngjashme të cilat mund të përdoren për të njejtin qëllim janë: Siemens Sentron PAC3200, Eaton Power Xpert Meter 4000, Fluke 1738 Power Logger, Delta Electronics PMU-960 etj.

Një nga qasjet tradicionale për matjen e energjisë është përdorimi i pajisjeve analoge për matjen e tensionit dhe rrymës, të cilat më pas shndërrohen në vlera digjitale për përpunim. Megjithatë, këto sisteme kanë kufizime në saktësinë e matjeve, mostrim jo të shpeshtë të vlerave dhe mungesë të ndërfaqeve për t'u manipulular nga ana e përdoruesit. Për këtë arsye, janë zhvilluar sisteme digjitale më të avancuara që përdorin shndërrues analog-digjital dhe procesorë të fuqishëm për të siguruar matje të saktë dhe të besueshme të energjisë.

Në këtë kontekst, ky punim ka për qëllim zhvillimin e një matësi të energjisë të bazuar në teknologjinë digjitale, i cili integron një ndarës tensioni, amplifikator dhe shndërrues analog-digjital për matjen e tensionit dhe rrymës, si dhe përpunimin e të dhënave në kohë reale. Problemi kryesor që trajtohet është sigurimi i një metode të besueshme dhe të saktë për matjen dhe monitorimin e konsumit të energjisë, duke minimizuar ndikimin e gabimeve të mundshme të shkaktuara nga ndryshimet në kushtet e punës.

Për zgjidhjen e këtij problemi, është propozuar një sistem që përdorë mikrokontrollerin AT89C51, i cili është përgjegjës për përpunimin e të dhënave dhe komunikimin me pajisjet e jashtme. Sistemi përfshin gjithashtu përdorimin e një ADC MCP3208 për shndërrimin e sinjalit analog në digjital dhe një EEPROM për ruajtjen e të dhënave. Ky projekt shërben si një zgjidhje inovative dhe efikase për monitorimin dhe menaxhimin e konsumit të energjisë, duke ofruar të dhëna të sakta dhe të besueshme në kohë reale, ndërkohë që mundëson ruajtjen afatgjatë të vlerave të konsumit të energjisë.

2. Përpunimi analog i tensionit dhe rrymës

2.1 Amplifikatori operacional TL072

TL072 është një prej komponentëve themelore që përdoret në përpunimin analog të sinjaleve në rastin e matësit të energjisë. Është një amplifikator operacional me përdorim të gjerë dhe me përformancë adekuate për aplikime si në këtë rast. Karakteristikat themelore të tij janë:

1. Zhvendosje e ultë e tensionit në hyrje, me vlerë maksimale 3 [mV], që minimizon gabimin në dalje duke e bërë të përshtatshëm për aplikime ku kërkohet amplifikim dhe vlera të sakta.
2. Rryma të vogla në hyrje, rreth 20 [nA], gjë që ulë ndikimin e komponentit në qarqe që kërkojnë impedance të lartë në stadin pasues, apo tërheqje të paktë të rrymës nga burimi i sinjalit, siç janë ndarësit e tensioneve apo sensorët.
3. Impedancë të lartë të hyrjes, rreth 10^{12} [Ω]
4. Brez të gjerë të tensioneve të funizimit nga ± 3 [V] deri ± 18 [V]

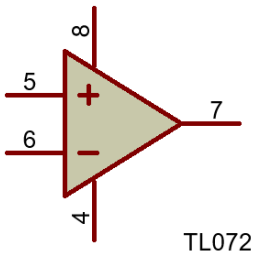
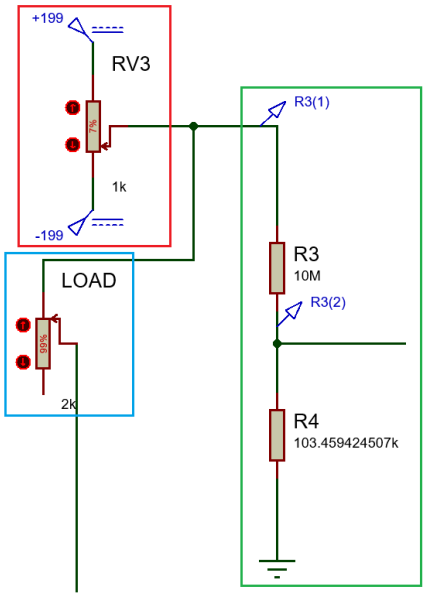


Fig. 2 – Amplifikatori operacional TL072

2.2 Matja e tensionit - Stadi hyrës



Vlerat e tensionit që maten mund të jenë të larta kështuqë nevojitet një qark dobësues i tensionit si stad hyrës i pajisjes në mënyre që vlerat e tensionit të jenë të përshtashme për përpunim nga pjesa digjitale. Për këtë qëllim përdoret ndarësi i tensionit, të cilit i asocion kuadranti i gjelbër.

Vlerat e tensionit në dalje varen nga tensioni në hyrje dhe nga vlerat e rezistorëve të vendosur. Llogaritja e vlerave të tensionit mund të bëhët me metoda klasike, duke e përdorur ligjin e parë dhe të dytë te Krichoffit. Në rastin më të thjeshtë më dy rezistorë, përcaktohet sipas relacionit më poshtë. [1]

$$V_o = \frac{R_b}{R_a + R_b} V_{in} \quad (1.1)$$

Fig. 3 - Stadi hyrës për qarkun e tensionit

Kuadranti i kuq në figurën 3 i korrospodon një “burimi” të tensionit me mundësi për ta ndryshuar, por vetëm për qëllime të simulimit, pasiqë një konfiguracion i tillë nuk mund të përdoret si burim i tensionit në kushte praktike për shkak të efiqiencës së ultë. Efekti i ngarkesës në “burim” nga ndarësi i tensionit nuk është përfillur. Ndërsa me ngjyrë të kaltër është treguar ngarkesa për të cilën bëhën matjet.

Efekti i ngarkesës

Para se sinjali të përcjellet në stadet tjera për përpunim analog dhe digjital duhet të shqyrtohet efekti i ngarkesës, që paraqitet vizualisht në figurën 4.

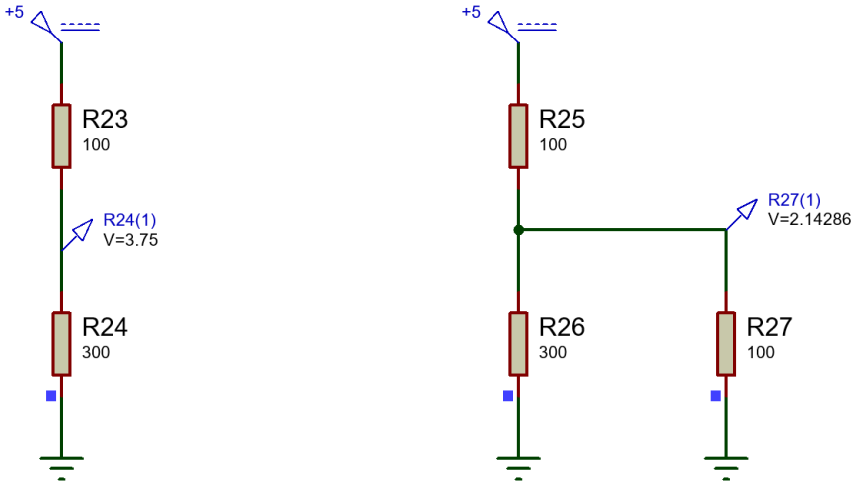


Fig. 4 – Ndarësi i tensionit pa ngarkesë dhe me ngarkesë

Në qoftë se rezistenca hyrëse e stadit pasues që lidhet më stadin hyrës është e vogël atëherë vlerat e tensionit në dalje të stadit hyrës ndikohen për shkak të efektit të ngarkesës, konkretisht për faktin se rezistenca hyrëse e stadit pasues lidhet paralel me ngarkesën R_b dhe rezistenca ekuivalente e rezistencave të lidhura paralelisht është çdoherë më e vogël se rezistenca më vlerën më të ultë, apo në rastin më të mirë, e barabartë me të.

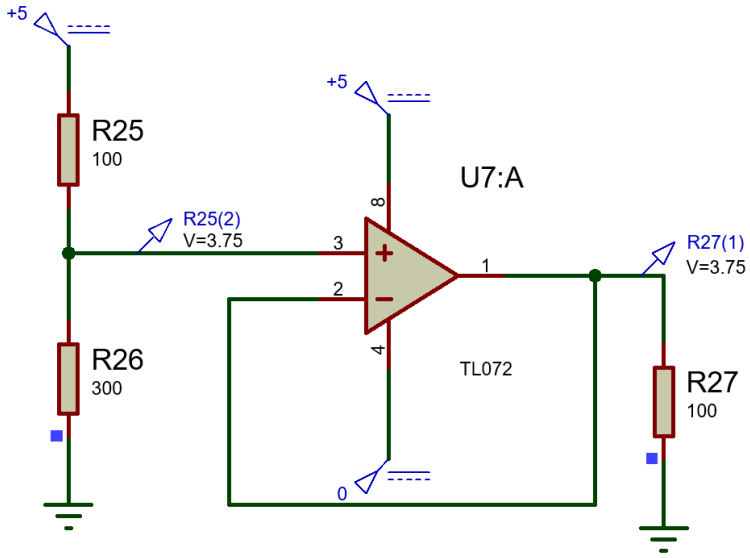


Fig. 5 – Anulimi i efektit të ngarkesës më percjellës të tensionit

Anulimi i efektit të ngarkesës mund të bëhët duke përdorur një amplifikator operacional më rezistencë hyrëse shumë të lartë [1], psh TL072 në konfiguracionin përcjellës i tensionit, siç është paraqitur në figurën 5.

Stadi hyrës i reprezentuar në figurën 1, për vlerat e burimit për brezin -199 [V] deri +199 [V] jep në dalje tension në brezin -2.048 [V] deri +2.048 [V].

2.3 Matja e tensionit – Përpunimi analog dhe shndërrimi i brezit

Për arsyet e përmendura më lartë në dalje të ndarësit të tensionit vendoset një përcjellës i tensionit, që është reprezentuar me kuadrantin e kuq në figurën 6. Në mes vendoset një mbledhës tensionit, që vlerës së sinjalit pas përcjellësit ia shton +2.048 [V] kështu duke dhënë në dalje sinjalin në brezin 0 [V] deri -4.096 [V] sipas relacionit:

$$V_o = -\left(\frac{R_f}{R_a}V_a + \frac{R_f}{R_b}V_b + \frac{R_f}{R_c}V_c\right) \quad (1.2)$$

Dhe në stadin e fundit të shndërruesit të brezit një invertues me vlerë të amplifikimit 1 e shndërron brezin në 0 [V] deri 4.096 [V], sipas relacionit:

$$V_o = -\frac{R_f}{R_a}V_a \quad (1.3)$$

Dalja e shndërruesit të brezit për tension lidhet me kanalin 0 të shndërruesit analog-digjital MCP3208.

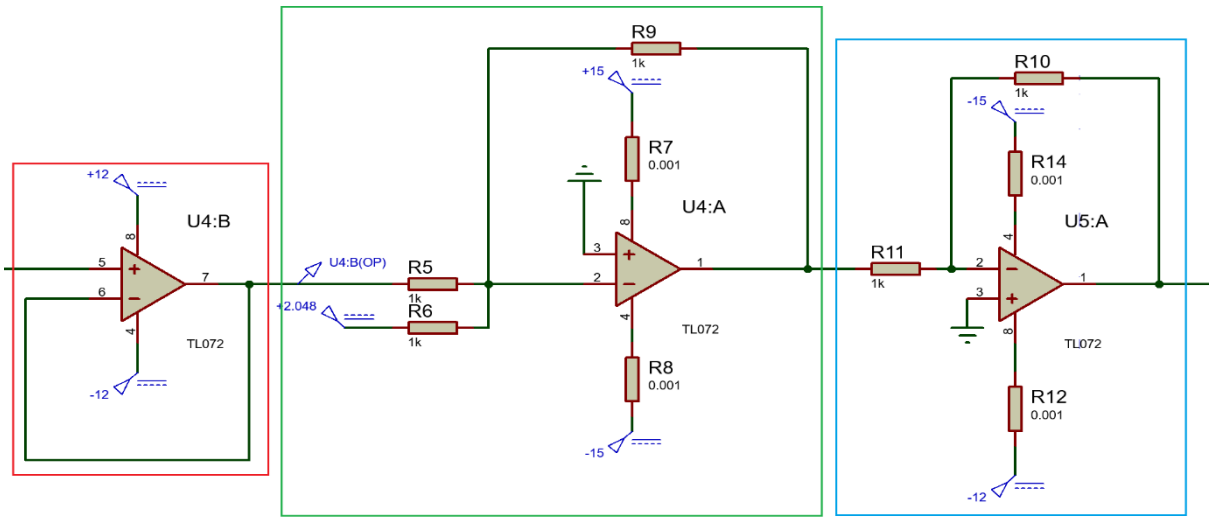


Fig. 6 – Shndërruesi i brezit për tension

2.4 Matja e rrymës – Stadi hyrës

Matja e rrymës nëpër një ngarkesë mund të bëhët më lidhjen e një rezistori me vlerë të ultë në seri me ngarkesën kështu duke formuar konfiguracionin e ndarësit të tensionit me dy rezistorë, si në figurën 7. Humbjet e tensionit në rezistorin internal të pajisjes me vlerë të ultë duhet të jenë minimale në mënyrë që ngarkesa të mos ndikohet nga pajisja për matje. Për shkak të sensitivetitetit të komponentëve të përzgjedhura për dizajnim, toleranca për humbje te tensionit në rezistencën e brendshme të është vendosur të jetë deri në 1%.

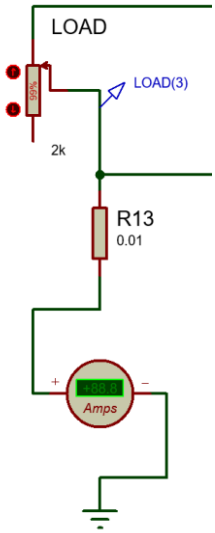


Fig. 7 – Stadi hyrës për qarkun e rrymës

Nëse humbjet e lejuara në rezistencën e brendshme të pajisjes matese janë deri 1% e vlerës se tensionit të burimit, atëherë nga formula 1.1 kemi:

$$V_o = 1\%V_i, \quad \frac{R_b}{R_a + R_b} = 0.01$$

Nëse rezistencën e brendshme të pajisjes e zgjedhim 0.01 [Ω], atëherë humbjet do të jenë 1% vetëm në rastin kur ngarkesa $R_a = 0.99$ [Ω], në rastet tjera kur vlera e ngarkesës është më e madhe se 0.99 [Ω], humbja e tensionit për shkak të pajisjes matëse do të jetë më e vogël se 1%.

Rezolucioni i shndërruesit analog-digjital MCP3208 të përdorur në dizajn është 1 [mV], që nënkupton se për vlerat e tensionit më të vogla se 0.001[V], shndërruesi nuk ka sensitivitet të majftushëm që t'i shoqëroj vlerë digjitale specifike në dalje të tij ashtuqë të dihet më saktësi vlera e tensionit në dalje të ndarësit të tensionit. Ndërsa vlera maksimale që mund të lexohet nga shndërruesi për shkak të kufizimeve harduerike dhe konfigurimit të jashtëm është 4.096 [V].

Sipas relacioni 1.1, kufiri i poshtëm 1 [mV] arrihet me ngarkesë 2 [$k\Omega$] për vlerë maksimale të tensionit:

$$0.001\ [V] = \frac{0.01\ [\Omega]}{R_a + \ 0.01\ [\Omega]} * 199\ [V]$$

Në këtë rast rryma do të jetë 100 [mA]. Për vlera më të mëdhaja të ngarkesës dhe rryma më të vogla se 100 [mA] duhet të përdoret amplifikues i tensionit në dalje të ndarësit apo të përdoret shndërrues analog-digjital me sensitivitet më të madh. Nëse sinjali i tensionit përforcohet për 100000 herë saktësia e matësit të rrymës do të jetë e rendit μA.

2.5 Matja e rrymës – Përpunimi analog dhe shndërrimi i brezit

Për shkak të efektit të ngarkesës stadi i parë duhet të jetë përcjellës i tensionit, me kuadrantin e gjelbër në figurën 8, sinjalit i shtohet +2.048 [V], kështu sinjali prej -2.048 [V] deri 0 [V] i asocion vlerave negative të rrymës për tension në brezin -199 [V] deri 0 [V], ndërsa për vlerat 0 [V] deri 199 [V] vlerat e shndërruara janë 0 [V] deri 2.048 [V]. Në dalje të mbledhësit vlerat dalin të invertuara, prandaj duhet të përdoret një invertues i sinjalit me amplifikim 1. Dalja e shndërruesit të brezit lidhet me kanalin 1 të shndërruesit analog-digjital MCP3208.

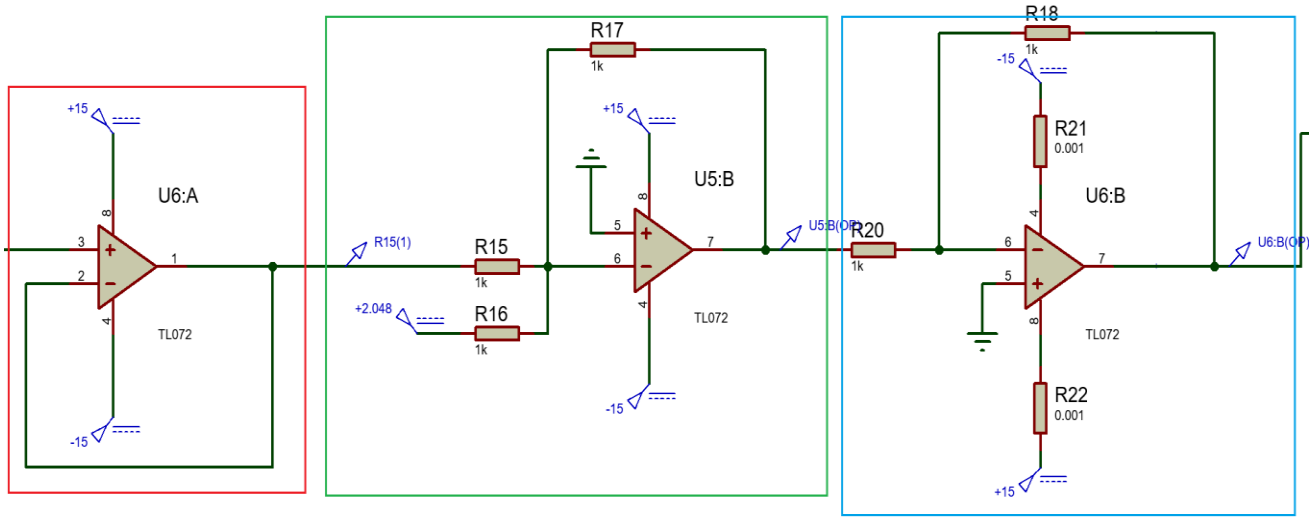


Fig. 8 – Shndërruesi i brezit për rrymë

3. Përpunimi digjital i tensionit dhe rrymës

3.1 Mikrokontrolleri AT89C51

AT89C51 është një mikrokontroller 8-bitësh prodhuar nga Atmel (tani pjesë e Microchip Technology), i përdorur gjerësisht në sistemet e integruara për shkak të fleksibilitetit, lehtësisë së përdorimit dhe kostos së ulët. Bazuar në arkitekturën 8051, AT89C51 është një mikrokontroller i besueshëm, ideal për aplikacione që kërkojnë kontroll të thjeshtë dhe përpunim të të dhënave, siç është në matësin e energjisë, sistemet e automatizimit dhe marrjen e të dhënave bazike.

Karakteristikat e tij përfshijnë:

- **Memorie ROM/EPROM**: AT89C51 ka 4KB memorie flash në çip, duke mundësuar ri-programimin e mikrokontrollerit gjatë zhvillimit. [7]
- **RAM**: Posedon 128 bajtë RAM në çip, mjaftueshëm për aplikacione të vogla që kërkojnë përpunim të të dhënave në kohë reale dhe ruajtje të tyre.
- **Pins I/O**: Ka 32 pina I/O të programueshëm, të shpërndarë në katër porte 8-bitëshe (P0, P1, P2, dhe P3), të cilat përdoren për funksione të ndryshme hyrëse dhe dalëse, siç janë kontrollimi i ekraneve, sensorëve dhe moduleve të komunikimit. [2]
- **Tajmerë dhe numërues**: Mikrokontrolleri ka dy tajmerë/numërues 16-bitësh, të cilët mund të përdoren për operacione të bazuara në kohë ose numërimin e eventeve, duke e bërë të përshtatshëm për monitorim në kohë reale dhe aplikime si matësi i energjisë. [2]
- **Komunikimi serik**: AT89C51 mund ta realizoj komunikimin serik përmes modulit të integruar UART (Universal Asynchronous Receiver-Transmitter). Kjo mundësi është veçanërisht e dobishme për krijimin e ndërfaqeve me pajisje periferike. [2]
- **Interraptët**: Ka pesë interrapte, duke mundësuar reagim në evente të ndryshme në kohë reale. [2]
- **Shpejtësia e klokut**: AT89C51 operon deri në shpejtësinë e klokut 24 MHz, e cila siguron fuqi të mjaftueshme për trajtimin e algoritmeve bazë dhe marrjes së të dhënave. [7]

Roli në matësin e energjisë

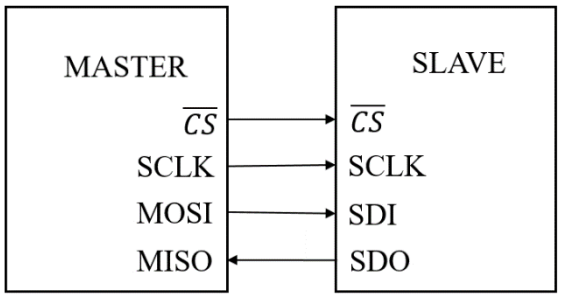
Në kuadër të këtij projekti të matësit të energjisë, mikrokontrolleri AT89C51 shërben si njësia qendrore për përpunimin e të dhënave të marrura shndërruesi analog-digjital MCP3208, i cili lexon vlerat e tensionit dhe rrymës. Pasi MCP3208 dërgon vlerat digjitale në AT89C51, mikrokontrolleri kryen llogaritjet e nevojshme për fuqinë dhe energjinë, duke përfshirë shumëzimin e tensionit dhe rrymës dhe akumulimin e energjisë gjatë kohës për fuqinë dhe energjinë, duke përfshirë shumëzimin e tensionit dhe rrymës dhe akumulimin e energjisë gjatë kohës.

AT89C51 përpunon këto të dhëna në kohë reale dhe i jep rezultatet në ekranin LCD. Për më tepër, mikrokontrolleri menaxhon komunikimin mes sistemit dhe EEPROM për ruajtjen e të

dhënave të energjisë gjatë kohës. Aftësitë e komunikimit serik të mikrokontrollerit përdoren gjithashtu për të komunikuar me pajisje periferike si MCP3208 përmes protokollit SPI dhe EEPROM FM24C256 përmes protokollit I2C.

3.2 Protokoli i komunikimit SPI

Serial Peripheral Interface (SPI) është një prej komunikimeve më të përdorur mes mikrokontrollerëve dhe qarqeve të integruara periferike si senzorëve, ADC, DAC, shiftregjistrave etj. SPI është lloj i komunikimit sinkron, full duplex. Shkëmbimi i të dhënave nga master në slave sinkronizohen në tehun ngritës apo rënës të klokut. Master dhe slave mund të dërgojnë të dhëna njëkohësisht. SPI në nivel fizik mund të realizohet me 3 tela ose me 4 tela, megjithëkëtë version i fundit është më shumë i përdorur.



Katër telat e pajisjeve që përdorin SPI, reprezentuar në figurën 9 i korrspodojnë 4 sinjaleve:

- Chip Select (CS)
- CLOCK (SCLK)
- Master out, slave in (MOSI)
- Master in, slave out (MISO)

Fig. 9 – Topologjia e SPI Master-Slave

Pajisja që gjeneron sinjalin e klokut quhet master. Transmetimi i të dhënave mes master dhe slave sinkronizohet me sinjalin e klokut të gjeneruar nga master. Pajisjet që përdorin komunikimin SPI mund të operojnë në frekuenca më të larta të klokut sesa ato që përdorin I2C. Megjithatë çdoherë duhet të përdoret dokumentimi përkatës i qarkut të integruar si referencë për përdorim adekuat. Në një topologji ku përdoret komunikimi SPI, mund të ekzistojë vetëm një master dhe një apo më shumë slave.

Sinjali Chip Select nga master përdoret për ta zgjedhur slave më të cilin do të realizohet komunikimi. Nëse dërgon HIGH e shkëput slave-in nga busi i SPI ndërsa nëse dërgon LOW, slave-i zgjedhet për të komunikuar. Kur ka më shumë së një slave në topologji, çdo çip ka një sinjal specifik, të dedikuar për komunikim më të.

MOSI dhe MISO janë telat (sinjalet) për shkëmbimin e të dhënave. MOSI përdoret për dërgimin e të dhënave nga master në slave, ndërsa MOSI përdoret për dërgimin e të dhënave prej slave në master.

Transmetimi i të dhënave

Që të filloj komunikimi SPI, master-i dërgon një sinjal për ta zgjedhur slave për komunikim, si më lartë, slave zgjedhet më sinjal LOW, prandaj pajisja kontrolluse master duhet të dërgoj 0 logjike. Gjatë komunikimit SPI, bitat shiftohet njëkohësisht dhe dërgohen në mënyre serike në telin MOSI/SDO dhe pranohen në telin MISO/SDI. Interface-i SPI lejon fleksibilitet në zgjedhjen e tehut rritës apo rënës të klokut që të përdoret për shiftimin e të dhënave në mënyrë serike. Ky është një element shumë më rëndësi i protokolit që duhet të respektohet gjatë implementimit të tij në një topologji master-slave.

Polariteti dhe faza e klokut

Në komunikimin SPI, master-i zgjedh polaritetin dhe fazën e klokut. Biti CPOL përcakton polaritetin e sinjalit të klokut gjatë gjendjes idle. Gjendja idle definohet si perioda kur CS nga HIGH kalon në LOW në fillim të komunikimit dhe kur CS nga LOW kalon në HIGH në fund të komunikimit. Biti CPHA përcakton fazën e klokut, varësisht nga biti CPHA, tehu rritës apo rënës i klokut përdoret për mostrim apo shiftim të bitave në telat për të dhëna. Masteri duhet të i zgjedh këta bita varësisht nga kërkesat e slave-it për komunikim. Varësisht nga CPOL dhe CPHA, ekzistojnë 4 mode të SPI, të specifikuara në tabelën 1.

Modi SPI	CPOL	CPHA	Polariteti i klokut në gjendjen idle	Faza e klokut e përdorur për mostrim dhe shiftim
0	0	0	LOW	Mostrimi në tehun rritës, shiftimi në tehun rënës
1	0	1	LOW	Mostrimi në tehun rënës, shiftimi në tehun rritës
2	1	0	HIGH	Mostrimi në tehun rënës, shiftimi në tehun rritës
3	1	1	HIGH	Mostrimi në tehun rritës, shiftimi në tehun rënës

Tabela 1 – Modet e SPI [4]

Konfiguracioni multi-slave

Më shumë se një slave mund të lidhen me një SPI master, siç është reprezentuar në figurën 9.

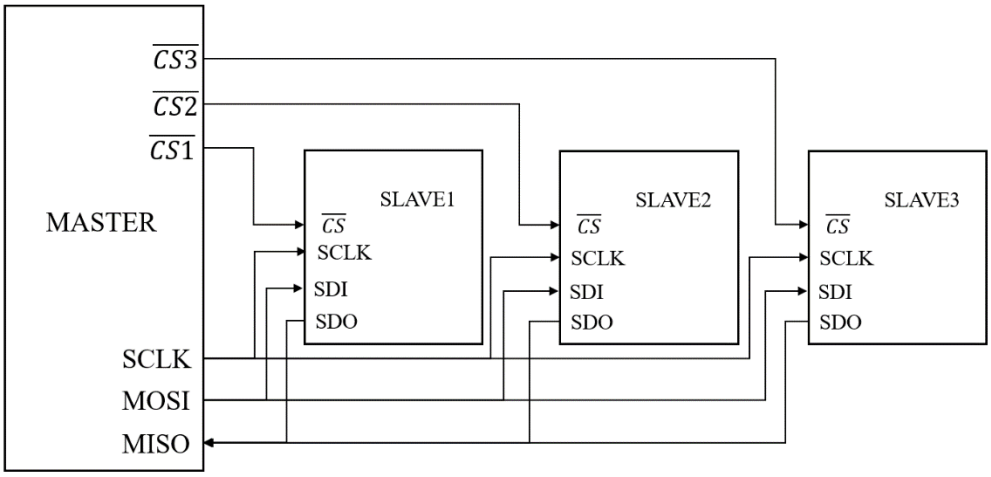


Fig. 10 – Topologjia master-multislave SPI

Në topologjin e paraqitur në figurën 10, cdo slave e ka të shoqëruar një Chip Select të veçant. Pasi që slave-i të jetë zgjedhur nga master-i, kloku dhe telat MISO/MOSI mund të përdoren. Nëse më shumë se një slave zgjedhet njëkohësisht atëherë të dhënat në MISO mund të korruptohen për shkak se nuk ka ndonjë mënyrë për ta dalluar se nga cili slave janë dërguar.

3.3 Shndërruesi A/D MCP3208

Shndërruesi A/D është një qark i integruar që e bën shndërrimin e një sinjali analog në sinjal digjital. Në rastin konkret përdoret MCP3208, është A/D 12-bit, që nënkupton se një mostër të sinjalit analog mund ta reprezentoj më një prej 4096 niveleve diskrete dhe digjitale. [8] Përparësia e këtij shndërruesi në krahasim me qarqet e familjes ICL710X qëndron pikërisht në numrin e mostrave për sekond. Nëse çipi operon në 5 [V] sinjali mund të mostrohet me shpeshtësi $f_{mostrimit}$ =100 000 mostra/sekond. Ndërsa frekuenca e klokut përcaktohet sipas formulës:

$$f_{klokut} = 20 \times f_{mostrimit} = 2 \text{ [MHz]}$$

Sipas kushtit të Nyquistit, për ta mostruar një sinjal kontinual në kohë, pa e humbur informacionin, shpeshtwsia e mostrimit duhet të jetë së paku dy herë frekuenca e sinjalit, matematikisht reprezentuar si:

$$f_{mostrimit} \geq 2 \times f_{sinjalit}$$

Prandaj, nëse sistemi do të përdoret për sinjale njëkahore dhe periodike apo edhe alternative (vetëm me modifikime sofuerike), atëherë matësi mund të përdoret pwr sinjale me frekuencë deri 50kHz. Ndërsa për sinjale me frekuencë të ultë (psh 50Hz), sistemi mund të përdoret edhe si matës i frekuencës (vetëm me modifikime softuerike).

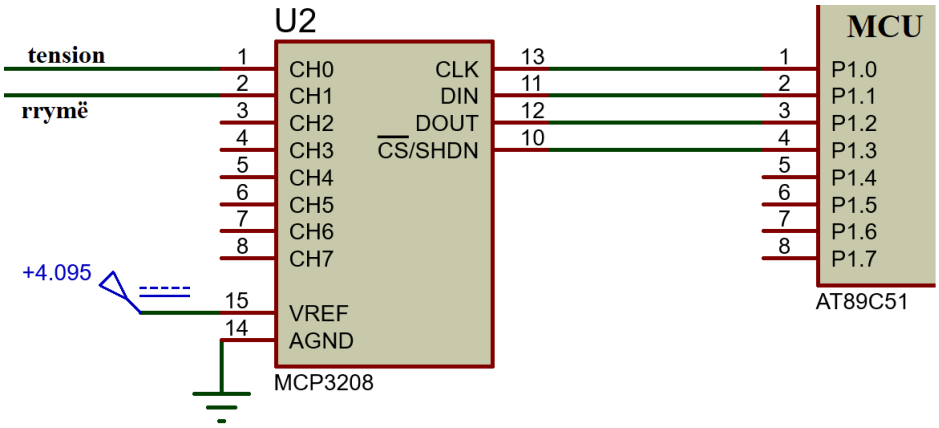


Fig. 11 – Shndërruesi A/D MCP3208

Tensioni i aplikuar në pinin V_{ref}, e përcakton brezin e tensionit më të cilin punon shndërruesi, gjithashtu e përcakton edhe diferencën mes niveleve diskrete fqinje apo sensitivitetin e shndërruesit ndaj ndryshimeve të sinjalit analog në hyrje. Në rastin konkret aplikohet tension prej 4.096 [V] në V_{ref}, që nënkupton se shkalla e sensitivitetit ndaj ndryshimeve të sinjalit në hyrje është 0.001V apo 1 [mV]. Për cdo vlerë analoge nga 0 [V] deri në 4.096 [V], shndërruesi jep një reprezentim digjital me 12 bita, i cili përcaktohet sipas formulës:

$$Kodi\ digjital\ në\ dalje = \frac{4096 \times V_{in}}{V_{ref}} \qquad (1.6)$$

Ku V_{in} është potenciali që aplikohet në hyrje të A/D në pinin CH0, i cili vjen nga dalja e shndërruesit të brezit për tension, apo në pinin CH1, i cili vjen nga dalja e shndërruesit të brezit për rrymë. Komunikimi i mikrokontrollerit 8051 me shndërruesin A/D realizohet sipas protokolit SPI (Serial Perhipheral Inverface), të reprezentuar në figurën 12.

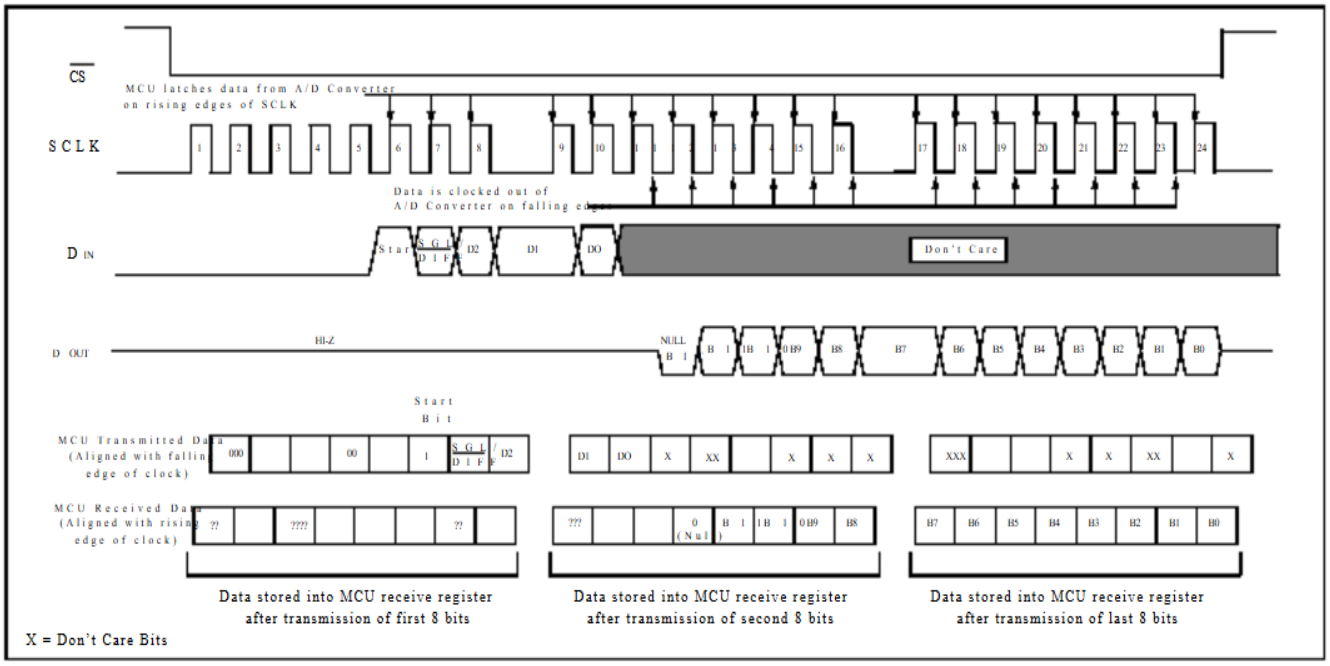


Fig. 12 – Komunikimi SPI për MCP3208, modi 0 (SCLK idles LOW) [8]

Që të filloj komunikimi nga mikrokontrolleri dërgohet LOW në pinin $\overline{CS}$, vendoset HIGH pini DIN nga dërgohet start-bit në MCP3208 prej 8051 në tehun rënës të klokut, kloku i MCP3208 kontrollohet nga mikrokontrolleri 8051 poashtu. Për t'u konfiguruar shndërruesi duhet të dërgohen Start-Bit, SGL/DIF. D2, D1 dhe D0.

Bitat e kontrollit				Konfiguracioni i hyrjes	Kanali i zgjedhur
SINGLE/$\overline{DIFF}$	D2	D1	D0		
1	0	0	0	me pikë referimi	Kanali 0
1	0	0	1	me pikë referimi	Kanali 1

Fig. 13 – Tabela për konfigurimin e shndërruesit A/D [8]

Për t'i lexuar vlerat e tensionit shndërruesi konfigurohet për ta përdorur kanali 0 ndërsa për rrymë kanali 1.

4. Arkitektura softuerike

4.1 Modulet dhe funksionet

Për të përshkruar arkitekturën e softuerit të matësit të energjisë, është e rëndësishme të përshkruhen komponentët e ndryshëm, përgjegjësitë e tyre dhe si ndërveprojnë me njëri-tjetrin. Më poshtë është një përmbledhje e arkitekturës, duke theksuar çdo modul softuerik, rolin e tij dhe komunikimin ndërmjet moduleve.

Siç është paraqitur në figurën 14, sistemi përbëhet nga disa module kyçe që ndërveprojnë me harduerin për të lexuar matjet, realizuar llogaritjet, shfaqur vlerat në LCD dhe ruajtur të dhënat në EEPROM. Menaxhimi qendror i sistemit bëhet nga **Menaxheri i Topologjisë** i cili integron funksionet e HAL dhe ekzekuton logjikën kryesore të aplikacionit.

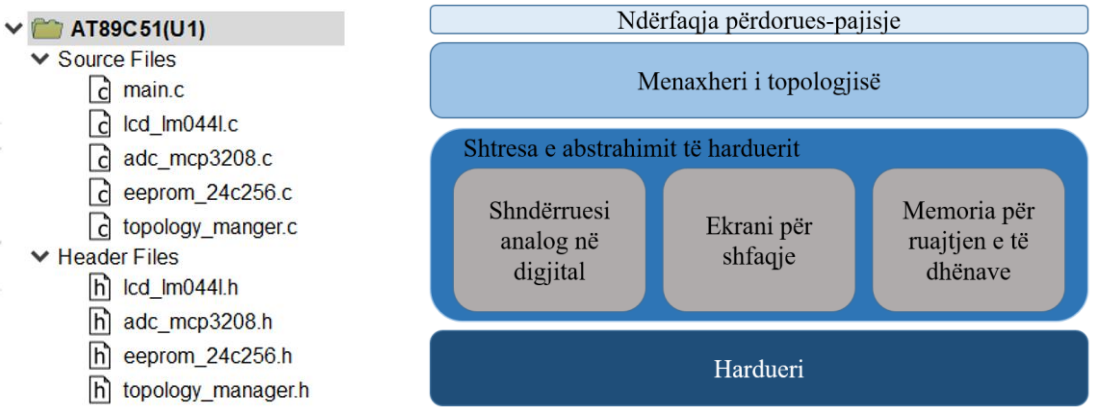


Fig. 14 – Organizimi dhe arkitektura e software-it për matësin e energjisë

Moduli – Menaxheri i topologjisë

Përgjegjësitë:

1. **Menaxhimi qendror**: Koordinon të gjitha funksionet në sistemin e matësit të energjisë. Lexon të dhënat nga sensorët, përpunon informacionin, bën llogaritjet dhe menaxhon logjikën për ruajtjen/shfaqjen.
2. **Kontrolli i rrjedhës**: Përcakton kur të aktivizohen leximet nga ADC, si dhe kur të procesohen ato, të ruhet rezultati në EEPROM dhe të azhurnohet ekrani LCD.
3. **Llogaritja**: Bën llogaritjen e fuqisë dhe energjisë bazuar në vlerat e tensionit dhe rrymës të lexuara nga ADC.
4. **Menaxhimi i LCD**: Përcakton kur dhe si të përditësohet ekrani LCD, si dhe formaton të dhënat për shfaqje.

Funksionet:

1. **measure_voltage(), measure_current(), calculate_power(), calculate_energy()**: Këto funksione ndërveprojnë me ADC dhe përdorin të dhënat për të llogaritur energjinë.

Moduli – Shndërruesi analog-digjital

Përgjegjësitë:

1. Moduli ADC është përgjegjës për komunikimin ndërmjet mikrokontrollerit dhe ADC (shndërruesit analog-digjital), duke siguruar vlerat e tensionit dhe rrymës që merren nga qarqet analoge dhe të shndërrohen në vlera digjitale të përpunueshme.

Funksionet:

1. **adc_delay()**: Ky funksion është përgjegjës për të siguruar një vonesë të nevojshme për të mundësuar që mikrokontrollori dhe ADC të jenë të sinkronizuar në frekuencë
2. **adc_spi()**: Ky funksion përdoret për të komunikuar me ADC përmes interfaces SPI dhe për të lexuar të dhënat digjitale nga një kanal specifik i ADC.

Moduli – Ekrani për shfaqje

Përgjegjësitë:

1. Moduli LCD është përgjegjës për menaxhimin e komunikimit me ekranin LCD të sistemit.

Funksionet:

1. **lcd_delay()**: Ky funksion krijon një vonesë të nevojshme për të mundësuar që mikrokontrollori dhe LCD të jenë të sinkronizuar në frekuencë.
2. **lcd_write_cmd()**: Ky funksion përdoret për të dërguar komanda të drejtpërdrejta në LCD. Siç mund të jetë komandë për kthimin e kursorit, vendosjen e rreshtit, ose komandat për konfigurimin e ekranit.
3. **lcd_write_data()**: Ky funksion përdoret për të dërguar të dhëna (në formën e një karakteri) në LCD për shfaqje.
4. **lcd_write_word()**: Ky funksion përdoret për të shkruar një fjalë (ose një varg karakteresh) në LCD.
5. **lcd_clr()**: Ky funksion është përgjegjës për pastrimin e përmbajtjes së ekranit LCD, duke fshirë çdo informacion të shfaqur dhe duke e përgatitur ekranin për të shfaqur të dhëna të reja.

Moduli – EEPROM

Përgjegjësitë:

1. **Menaxhimi i komunikimit me EEPROM**: siguron ndërfaqen për menaxhimin e komunikimit mes mikrokontrollerit dhe EEPROM.
2. **Protokoli I2C**: Përgjegjës për inicializimin, startimin dhe përfundimin e komunikimit I2C me EEPROM, duke përdorur komandat për të shkruar dhe lexuar të dhëna.

Funksionet:

1. **eeprom_delay()**: Ky funksion krijon një vonesë të nevojshme për të siguruar që mikrokontrolleri dhe EEPROM të jenë të sinkronizuar në frekuencë.
2. **i2c_init()**: Ky funksion inicializon komunikimin I2C midis mikrokontrollerit dhe EEPROM. Ai përgatit kushtet e nevojshme për të mundësuar komunikimin e saktë mes pajisjes së mikrokontrollerit dhe EEPROM, duke siguruar se të dy pajisjet janë të gatshme për komunikim.
3. **i2c_start()**: Ky funksion fillon komunikimin I2C duke dërguar një sinjal start. Ky sinjal sinjalizon fillimin e një sesioni komunikimi mes mikrokontrollerit dhe EEPROM. Përdoret për të nisur procesin e shkruarjes ose leximit të të dhënave nga EEPROM.
4. **getAck()**: Ky funksion merr acknowledge bit nga EEPROM pas dërgimit të një komande nga mikrokontrolleri. Ky bit përdoret për të verifikuar nëse komandat janë pranuar dhe ekzekutuar me sukses nga EEPROM.
5. **eeprom_write_page()**: Ky funksion përdoret për të shkruar një numër bajtësh në EEPROM.
6. **eeprom_write_byte()**: Ky funksion përdoret për të shkruar një bajt të vetëm në EEPROM.
7. **clear_eeprom()**: Ky funksion është përgjegjës për pastrimin e të dhënave në EEPROM, duke fshirë të gjitha të dhënat e ruajtura.
8. **i2c_stop()**: Ky funksion përfundon komunikimin I2C, duke dërguar një sinjal stop për të mbyllur sesionin e komunikimit midis mikrokontrollerit dhe EEPROM.

Moduli i ndërfaqes përdorues-pajisje përfshin butonin që përdoret për të ruajtuar të dhënat në eeprom dhe moduli harduerik i reprezentuar në figurën 15 është shpjeguar detajisht ne seksionet më lartë. Modularizimi i çdo komponente (ADC, EEPROM, LCD) është i abstrahuar në modul të vetin, duke e bërë sistemin fleksibil dhe të mirëmbajtshëm. Menaxheri i topologjisë kryen operacionet e nivelit të lartë, ndërsa komponentët e **HAL** janë përgjegjës për komunikimin e drejtpërdrejtë me harduerin. Kjo redukton kompleksitetin dhe bën më të lehtë përditësimin ose zëvendësimin e moduleve harduerike pa ndikuar në logjikën kryesore.

4.2 Algoritmet themelore

Në çdo sistem softuerik të avancuar, algoritmet luajnë një rol kyç në përcaktimin e mënyrës se rrjedhës logjike, si përpunohet informacioni dhe si sillet sistemi në përgjithësi. Në këtë seksion, tregohen algoritmet kryesore të implementuara në sistemin e matësit te energjisë, të sipas të cilave menaxhohet monitorimi, matja dhe ruajtja e parametrave kyç si tensioni, rryma, fuqia dhe energjia. Këto algoritmë janë dizajnuar për të reaguar në kohë reale, të realizuara me mekanizmin e interruptit që mikrokontrolleri e posedon. Sistemi është i strukturuar që të funksionoj vazhdimisht pa ndërprerje, duke siguruar leximе të shpeshta dhe duke ruajtur integritetin e të dhënave gjatë gjithë kohës.

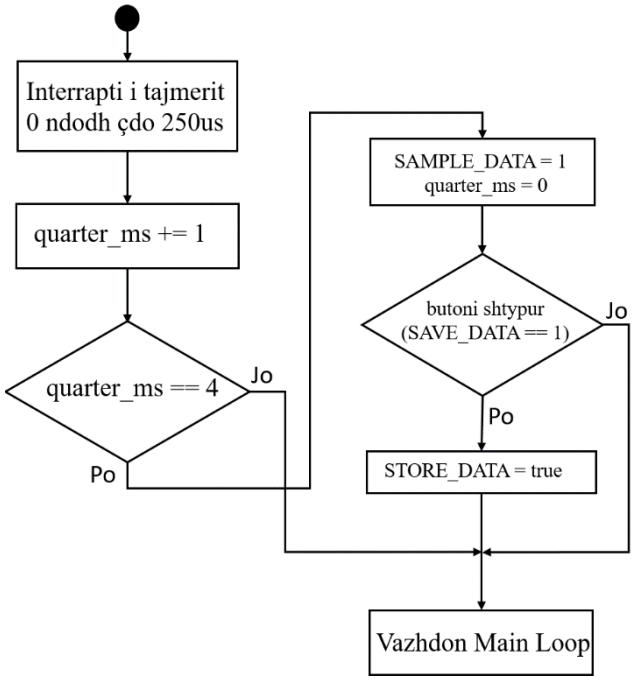


Fig. 15 – Algoritmi i interruptit timer 0 në modin 8-bit autoreload

T0_ISR është një interrupt rutinë që shkaktohet çdoherë që T0 overflow. Ky funksion ekzekutohet automatikisht nga mikrokontrolleri kur interrapti i tajmerit shkaktohet.Variabla **quarter_ms** inkrementohet cdo interrupt, në rastin konkret interrupti ndodh cdo 250 [us], dhe resetohet cdo 1ms. [3] Variabla **ms** ruan numrin e milisekondave që kanë kaluar dhe resetohet cdo 1s. Flagu SAMPLE_DATA përdoret për të njoftuar pjesen tjetër të programit se kur duhet të mostrohen vlerat e tensionit dhe rrymës. Flagu STORE_DATA përdoret për të ruajtur vlerën e buttonit që përdoret për ruajtjen e të dhënave aktuale, përmbajtja e tij propagohet në flagun

SAVE_DATA që më vonë përdoret në pjesën tjetër të programit për ta thirrur funksionin që mundëson ruajtjen e të dhënave në EEPROM.

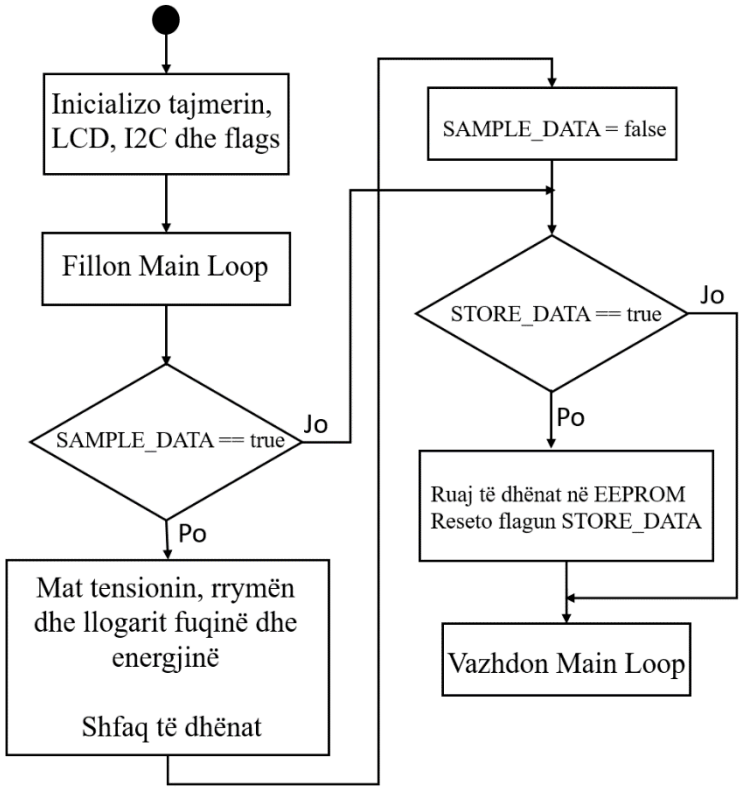


Fig. 16 – Algoritmi i funksionit main

Ky algoritëm reprezenton funksionalitetin e matësit të energjisë që në mënyre periodike i matë, shfaq dhe ruan parametrat e tensionit, rrymës, fuqisë dhe energjisë. Sistemi është i dizajnuar të punoj në infinite loop, duke monitoruar në mënyrë kontinuale kushtet për mostrim dhe përpunim të parametrave. Sistemi inicializohet duhet i përgaditur njësitë harduerike periferike për të komunikuar dhe pasi të jetë inicializuar, varësisht nga vlera e flagut SAMPLE_DATA që përcaktohet nga algoritmi më lartë, mat tensionin dhe rrymën dhe llogarit fuqinë dhe energjinë. Ruajta e rezultateve varet nga vlera e flagut STORE_DATA që poashtu përcaktohet sipas algoritmit më lartë. Programi është i strukturuar që të punoj në kohë reale duke monitoruar dhe ruajtur parametrat elektrik të lartë përmendur.

4.3 Llogaritja e fuqisë dhe energjisë

Pjesa themelore e arkitekturës softuerike, ashtu si është pështkruar edhe në seksionet paraprake është menaxheri i topologjisë, kodi përkatës mundëson leximin e vlerave të tensionit dhe rrymës të matura nga qarqet periferike ndihmëse dhe llogarit fuqinë dhe energjinë.

Funksioni measure_voltage() lexon vlerat e tensionit nga shndërruesi analog në digjital, duke përdorur protokolin e komunikimit SPI. Funksioni adc_spi(kanali_0) kthen një numër unsigned int që përdoret për ruajtjen e një vlerë 12 bitëshe të dërguar nga shndërruesi analog-digjital. Kjo vlerë reprezenton tensionin e matur në hyrje, e cila duhet të shndërrohet në vlerë reale të tensionit. Nëse vlera e tensionit është më e madhe ose baras me 2048, nënkuptohet se vlera e tensionit është pozitive, kjo rrjedh nga dizajni i shndërruesit të brezit në stadin analog. Vlera reale e tensionit pas normalizimit ndaj vlerës referente 2.048 [V] dhe faktori të kushtëzuar nga qarku analog 97.65625 ruhet në një variabël voltage_volt_f të tipit float. Nëse vlera është më e vogël se 2048, atëherë nënkuptohet se vlera e tensionit është negative dhe aplikohet e njejta formulë për normalizim dhë shndërrim të vlerës se lexuar në vlerë reale.

```
void measure_voltage(){

    lcd_delay_clr();
    unsigned int voltage_volt = adc_spi(CHANNEL_0);

    if(voltage_volt >= 2048){
        is_neg_voltage = 0;
        voltage_volt_f = (((float)voltage_volt/1000.0) - 2.048)*97.65625;
    }else{
        is_neg_voltage = 1;
        voltage_volt_f = (((((float)voltage_volt)/1000.0) - 2.048)*(-1))*97.65625;
    }

    format_print_voltage();
}
```

Fig. 17 – Funksioni për leximin e vlerës së tensionit

Funksionit measure_current() punon në mënyrë të ngjashme, lexon vlerën e rrymës duke përdorur kanalin 1 të shndërruesit. Vlera e dërguar nga shndërruesi current_amp procesohet ngjashëm poashtu, për vlera më të mëdha se 2048, rryma ka vlerë pozitive dhe pas normalizimit vlera reale e rrymës ruhet në variablën current_amp_f të tipit float. Funksioni format_print_current () e formaton vlerën e rrymës në një numër me 3 shifra para presjës dhe 2 shifra pas presjës dhe e printon në LCD.

```
void measure_current(){
    lcd_delay_clr();
    unsigned int current_amp = adc_spi(CHANNEL_1);

    if(current_amp >= 2048){
        is_neg_current = 0;
        current_amp_f = (((float)current_amp/1000.0) - 2.048)*100.0;
    }else{
        is_neg_current = 1;
        current_amp_f = (((((float)current_amp)/1000.0) - 2.048)*(-1.0))*100.0;
    }

    format_print_current();

}
```

Fig. 18 – Funksioni për leximin e vlerës së rrymës

Funksioni calculate_power() shumëzon vlerat e variablave voltage_volt_f dhe current_amp_f për ta llogaritur fuqinë në ëat dhe ruan në variable power_wat_f. Sipas formulës:

$$Fuqia\ (W) = Tensioni\ (V) \times Rryma\ (A) \quad (1.7)$$

Energjia llogaritet duke integruar fuqinë përgjatë kohës, sipas formulës:

$$Energjia\ (Wh) = \int_0^t Fuqia\ (W)(dt) \quad (1.8)$$

Ku dt varet nga shpeshtësia e mostrimit të sinjalit nga mikrokontrolleri. Nëse vlerat tensionit, rrymës dhe fuqisë mostrohen çdo 1 ms, atëherë dt = 1/3600 000 (h).

```
void calculate_energy(){
    energy_wath_f = energy_wath_f + (power_wat_f*(1.0/3600000.0));
    format_print_energy();

}
```

Fig. 29 – Funksioni për llogaritjen e vlerës së energjisë

Variablës së tipit float në të cilën ruhet energjia në Wh çdo 1 ms i shtohet vlera e energjisë së llogaritur sipas formulës më lartë.

5. Shfaqja e vlerave

5.1 LCD LM044L

LCD LM044L është një ekran i thjeshtë që zë 20 karaktere në secilin nga 4 rreshtat. Komunikon me mikrokontrollerin duke përdorur një lidhje paralele 8-bitëshe që i lidh pinat për të dhëna të LCD dhe portin P0 të mikrokontrollerit. Pasi pinat e Portit P0 nuk kanë pull-up rezistorë të brendshëm (fig. 20), janë përdorur pull-up rezistorë të jashtëm për të siguruar nivel të dëshiruar të tensionit në pinat përkatës.

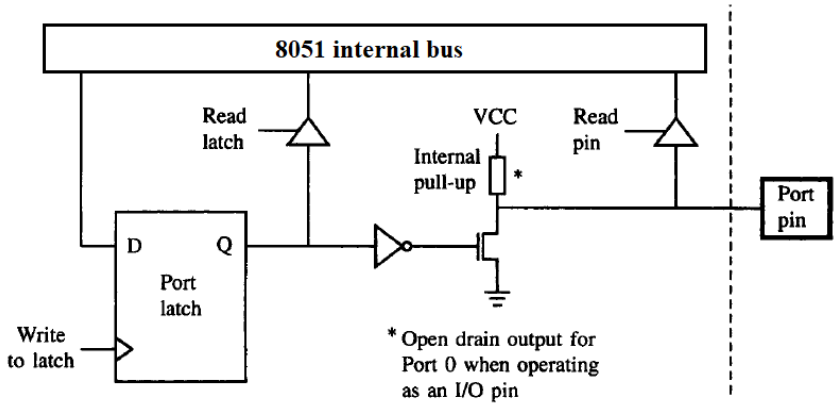


Fig. 20 Qarku për I/O pins [2]

Kontrolli i LCD

Pin-i RS është lidhur në P2.0 të mikrokontrollerit. Ky pin përdoret për të zgjedhur nëse dergojmë komandë apo të dhëna në LCD. Kur RS është LOW, LCD-ja interpreton të dhënat në D0-D7 si komanda, ndërsa kur RS është HIGH, të dhënat i interpreton si karaktere dhe i shfaq në ekran.

Pin-i E është lidhur në P2.1. Pin-i Enable përdoret për të lejuar që të dhënat të kalojnë nga busi në pjesën e brendshme të LCD. Konkretisht kjo bëhët në tranzicionin nga HIGH në LOW.

Pin-i RW është lidhur në GND. Kur në të aplikohet LOW nënkupton që në LCD vetëm do të shkruajmë dhe nuk do të lexojmë nga regjistrat internal të LCD-së.

Mikrokontrolleri dërgon komanda për detyra si vendosja e pozicionit të kursorit, kontrollimi I rrjedheës së shfaqjes, pastrami i ekranit etj. Të cilat janë të paraqitura në tabelën 1 më poshtë. Vlerat që shfaqen në LCD janë rezultat i matjeve në kohë reale të cilat përditsohen në mënyre kontinuale. Ky konfiguracion mundëson përdoruesit të monitorojnë me lehtësi konsumimin e energjisë dhe parametrat e tjerë të rëndësishëm në një format të thjeshtë siç është paraqitur në figurën 21.

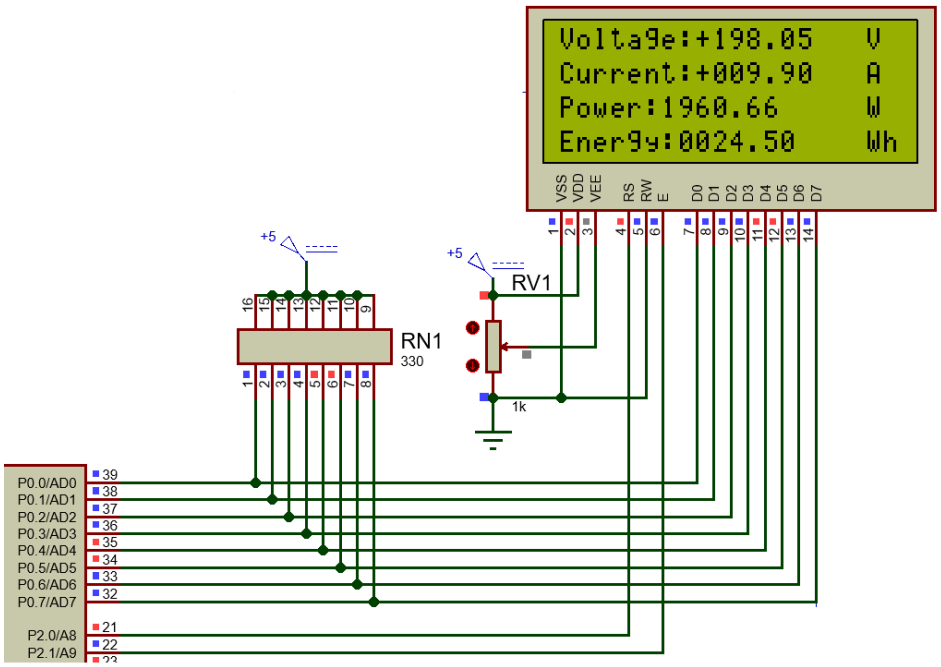


Fig. 21 – Konfiguracioni i display-t

Inicializimi i LCD-së apo komandat e para që duhet të dërgohen pasi të ketë filluar komunikimi janë, 0x38 që përcakton llojin e interface-it, 0x0C që e bën kursorit të padukshëm kështu duke mos u verejtur gjatë përditsimit dhe 0x80 e pozicionon kursorin (e padukshëm) të lokacioni përkatës.

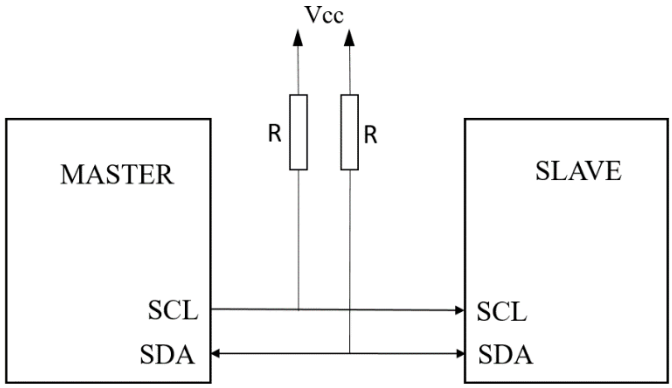
Komanda	Hex
Funksioni (ndërfaqe 8-bit, 4 linja, 5*7)	0x38
Funksioni (ndërfaqe 8-bit, 2 linja, 5*7)	0x30
Funksioni (ndërfaqe 4-bit, 4 linja, 5*7)	0x28
Funksioni (ndërfaqe 4-bit, 2 linja, 5*7)	0x20
Lëviz pamjen një karakter në të djathtë (të gjithë rreshtat)	0x1E
Lëviz pamjen një karakter në të majtë (të gjithë rreshtat)	0x1B
Home (vendos kursorin të pozita e karakterit lartë-djathtë)	0x02
Lëviz kursorin një karakter majtas	0x10
Lëviz kursorin një karakter djathtas	0x14
Apliko kursorin me nën-vijë	0x0E
Apliko kursorin bllok dhe blinkues	0x0F
Bëje kursorin të padukshëm	0x0C
Fsheh pamjen (pa e fshirë përmbajtjen)	0x0B
Rikthe pamjen (me kursor të fshehur)	0x0C
Fshijë pamjen	0x01
Vendos kursorin në pozitën e (adresës DDRAM)	0x80+
Vendos pointerin te karakteret e gjeneruara (adresa CG RAM)	0x40+

Tabela 2 - Komandat e LCD LM044L [9]

6. Ruajta e të dhënave

6.1 Protokoli i komunikimit I2C

I2C (Inter-Integrated Circuit) është një protokol i komunikimit serik half-duplex, multi-slave, multi-mater që mundeson lidhjen dhe komunikimin e shumë pajisjëve vetëm me 2 tela. Përdoret shpesh në komunikime mes mikrokontrollerave dhe njësive periferike si senzorët, EEPROM, ADC, DAC etj. I2C është i dizajnuar për distancë të shkurtëra të komunikimit, zakonisht brenda të njëjtit qark apo pllakë.



Dy telat e pajisjeve që përdorin I2C, reprezentuar në figurën 22 i korrspodojnë 2 sinjaleve:

- SCL
- SDA

Fig. 22 – Topologjia master-slave e protokolit I2C

SDA (Serial Data Line) është teli që përdoret për shkëmbimin e të dhënave në mes master dhe slave. Të dhënat transmetohen në bita, dhe gjendja e SDA mund të ndryshoj vetëm kur SCL është LOW (idle state).

SCL (Serial Clock Line) është sinjali i klokut i gjeneruar nga master. Kloku përdoret për sinkronizimin e shkëmbimit te të dhënave në telin SDA. Cdo bit transferohet me një puls të klokut.

Pull-up rezistorët janë të nevojshëm për të siguruar vlerat HIGH të SDA dhe SCL kur asnjë pajisje në topologji nuk i ka përtokezuar për të komunikuar, kështu duke indikuar që busi është i lirë të përdoret. Vlerat tipike të tyre janë 4.7 [$k\Omega$]deri në 10 [$k\Omega$]. [2]

Transmetimi i të dhënave

Cdo shkëmbim të dhënave në busin I2C fillon me START dhe terminohet me STOP. Tranzicioni prej HIGH në LOW në telin SDA përderisa SCL është HIGH definohet si kusht START. Tranzicioni prej LOW në HIGH në telin SDA përderisa SCL është HIGH definohet si kusht STOP, siç është paraqitur në diagramin kohor në figurën 23.

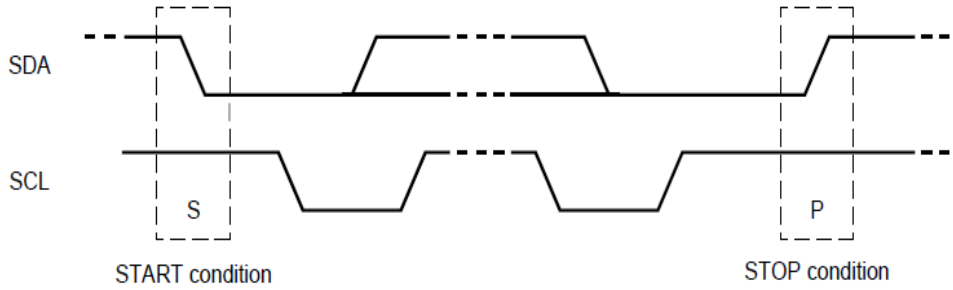


Fig. 23 – Kushtet për fillim/terminim të sesionit I2C [5]

Cdo njësi e dërgimit në busin I2C duhet ti ketë 8 bita. Numri i bajtave që mund të dërgohen pasi të ketë filluar sesioni i komunikimit është i pakufizuar, përveq nëse pajisja me të cilën komunikon ka kufizime për numrin e bajtëve. Cdo bajt duhet të pasohet me një ACK bit. Të dhënat transferohet me MSB së pari. Nëse slave nuk mund ta pranoj një bajt deri sa të përformohet ndonjë operacion internal i tij, mund ta mbaj klokun SCL LOW që ta forcoj master-in të pres.

Acknoëledge (ACK) është një bit që dërgohet pas çdo bajti të pranuar, kjo mundëson që slave të sinjalizoj masterin se bajti është pranuar me sukses dhe mund të vazhdohet me dërgimin e bajtit tjetër. Masteri i gjeneron të gjithat pulset e klokut, duke përfshirë edhe pulsing e nëntë për ta lexuar bitin ACK ose NACK. Nëse SDA qëndron HIGH gjat pulsit të nëntë, atëherë bajti nuk është pranuar, në këtë rast masteri dërgon STOP për ta abortuar transferrin, ose një START për ta rifilluar transferin e ri.

Pasi të jetë dërguar START dhe të ketë filluar komunikimi, dërgohet adresa e slave-it. Adresa është 7 bit e gjatë dhe pasohet nga biti i 8, i cili përcakton kahun e rrjedhës së të dhënave (R/W). Nëse dërgohet LOW, indikon që masteri do të shkruaj në slave, ndërsa nëse dërgohet HIGH, masteri kërkon të lexoj nga slave-i.

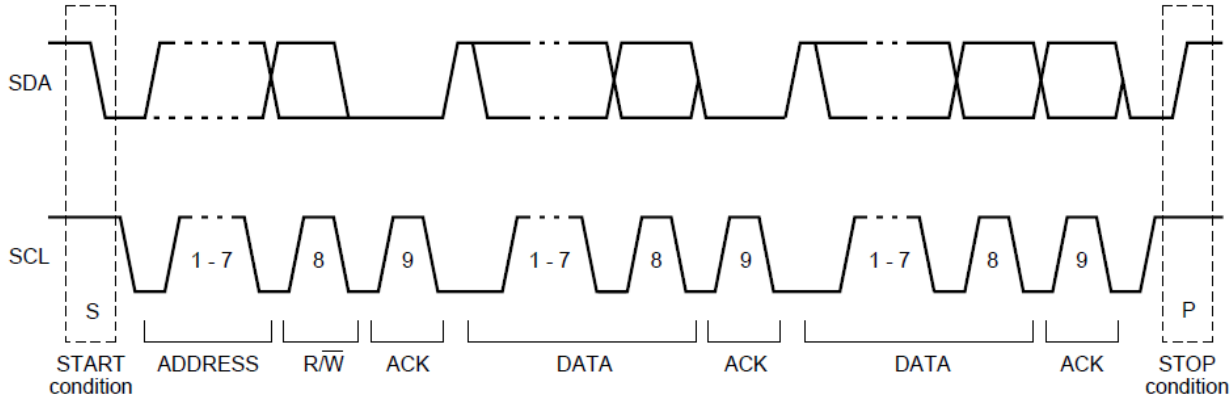


Fig. 24 – Diagrami kohor i sessionit të plotë I2C [5]

Mëqenëse numri i bitave më të cilet mund të adressohet nja slave në topologji është 7 atëherë numri maksimal i slave-ve që mund të përmbaj konfiguracioni është 128.

6.2 EEPROM FM24C256

Sistemi poashtu është i dizjanuar që të i ruaj matjet elektrike të tensionit dhe rrymës, dhe vlerave të llogaritura të fuqisë dhe energjisë. Vlerat maten në menyrë të vazhdueshmë, por ruhen vetëm pasi të jetë shtypur buttoni SAVE_DATA, reprezentuar në figurën 25. Ndërsa ledi i kuq ndezet në rastin kur memoria ka ndonjë problem të pariparushëm hardëerik apo ndonjë sjellje të pa pritur nga mikrokontrolleri gjatë komunikimit. Të dhënat ruhen në EEPROM eskternal për lexim apo analizë të mëvonshme.

EEPROM-i FM24C256 është një memorie 256 kilo bitëshe (32 KB), është e organizuar si 32768 bajtë. Cdo lokacioni i korrospodon një bajt, të dhënat shumë bajtëshe si numrat me floating point mund të ruhen si sekuencë e bajtëve në memorie. [10]

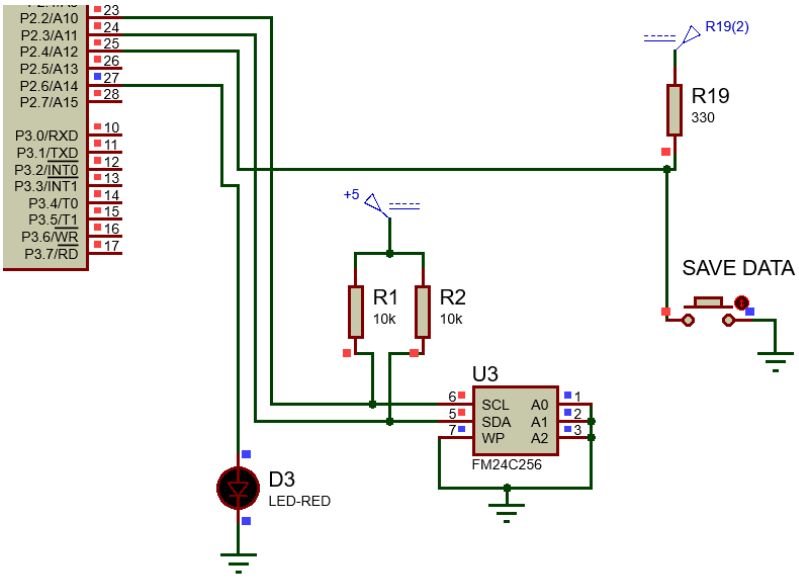


Fig. 25 – Konfiguracioni harduerik i ruajtes së të dhënave

Operacionet për shkrim dhe lexim realizohen pasi mikrokontrolleri dërgon frames të protokolit I2C. Operacionet për shkrim pasohen me kohën e ciklit të shkrimit prej 5 [ms] për byte. Bus-i i I2C lejon komunikim sinkron të dyanëshëm duke përdorur pinin SDA (Serial Data) për bartjen e të dhënave mes mikrokontrollerit dhe EEPROM-it dhe SCL (Serial Clock) përdoret për dërgimin e pulseve të klokut nga mikrokontrolleri në memorie për ta sinkronizuar komunikimin. Cdo session i komunikimit duhet të filloj me kushtin START dhe të përfundoj me kushtin STOP, ndërsa për shkëmbimet gjatë sessionit përdoret biti ACK për validim. Pini WP (Write Protection) përdoret për specifikimin e operacioneve që mund të kryhen në memorie, nëse WP lidhet me HIGH, operacioni për shkrim në memorie nuk mund të realizohet. Operacioni për lexim mund të përformohet pavarsisht lidhjes se pinit WP. Për EEPROM 4 bitat e parë the bajtit për kontroll janë 1010 për lexim dhe shkrim, pastaj këta pasohen me bitat A2, A1 dhe A0 dhe biti i fundit ne bajtin e kontrollit përdoret për ta determinuar operacionin që do të kryhet (lexim/shkrim). HIGH për lexim, ndërsa LOW për shkrim, si në figurën 26.

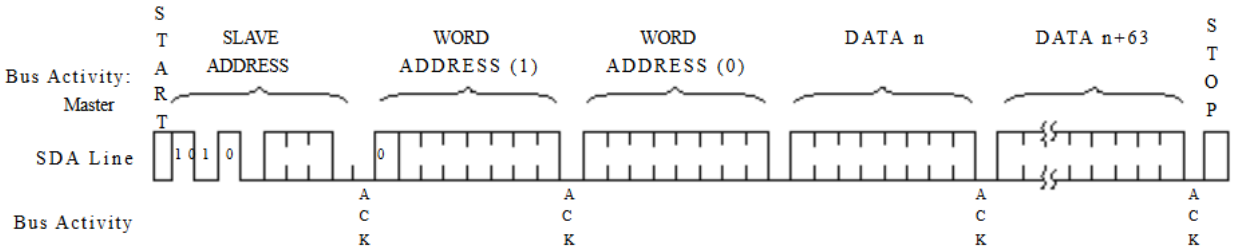


Fig. 26 – Frame i sessionit për komunikim I2C me FM24C256 [10]

Pinat A0, A1 dhe A2 që në rastin konkret janë lidhur LOW e specifikojnë adressën e EEPROM (slave) në rast të shumë pajisjeve në bus-in e njejtë, meqensë 3 pina përdoren për specifikim atëherë numri maksimal i pajisjeve të tilla që mund të vendosen në bus-in e njejtë me adresa të ndryshe është 8. Pas dërgimit të adresës se slave dhe biti R/W, dy bajtat pasues që dërgohen i korrspodojnë adresës 15-bitëshe (bit parë nga 16 është gjithmonë zero për këtë memorie) të lokacionit prej të cilit mikrokontrolleri do ta shkruaj të dhënat që pasojnë si bajta tjerë në mënyrë sekuenciale. Sessioni përfundon me STOP.

Për cdo matje të tensionit, rrymës, fuqisë dhe energjisë përdoren vlerat floating-point si format për ruajtje, cdo vlerë floating-point i zë 4 bajtë (32 bitë). Cdo matje ruhet si sekuencë e 16 bajtëshe me renditje: tension, rrymë, fuqi dhe energji, siç është reprezentuar në figurën 27.

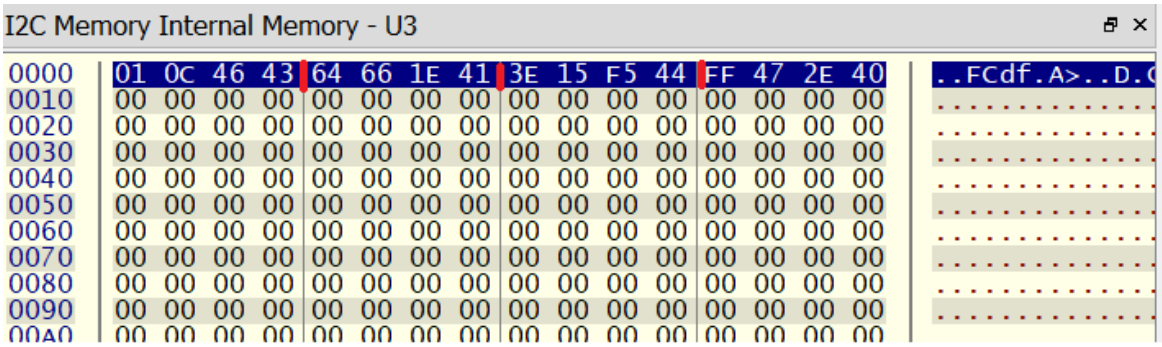


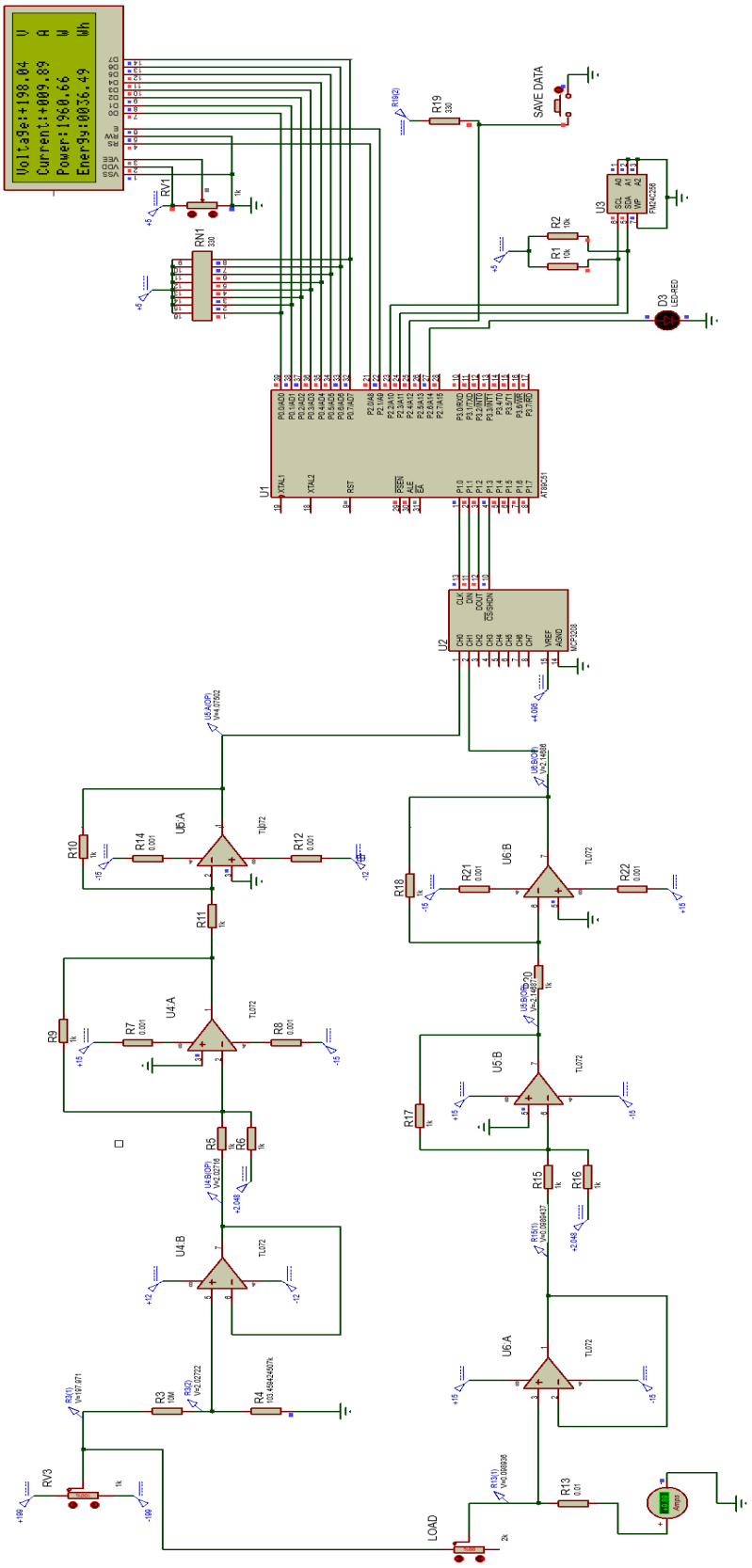
Fig. 27 – Vlerat e një matje në memorie

Vlerat e ruajtura në memorie të reprezentuara në figurën 17 pasi të shndërrohen në numra me floating-point dhe të printohen kanë këto vlera:

- Tension, 0x010C4643 = 198.047 [V]
- Rrymë, 0x64661E41 = 9.9 [A]
- Fuqi, 0x3E15F544 = 1960.66 [W]
- Energji, 0xFF472E40 = 2.723 [Wh]

Për deshifrimin e përmbajtjës së memories në këtë mënyrë mund të përdoret kodi i bashkangjitur në shtojcë.

Skema


Voltage:+190.04 V
Current:+009.89 A
Power:1960.66 W
Energy:0036.49 Wh
SAVE DATA
LOAD
U1
U2
U3
U4:A
U4:B
U5:A
U5:B
U6:A
U6:B
RV1
RV3
RN1
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
D3
LED-RED
TL072
AT89C51
MCP3208

7. Qarqet e integruara

Qarqet e integruara që përdoren për matjen e energjisë kanë rëndësi të lartë për shkak se e lehtësojne ndërtimin e pajisjeve komerciale për matjen e energjisë, duke zvogëluar numrin e komponentëve diskrete që duhet të përdoren në dizjan dhe ndërtim, poashtu mbështesin edhe ndërfaqe për dërgim te të dhënave të cilat mund të shfaqen dhe ruhen nga pajisje tjera perferike. Disa nga qarqet e integruara të tilla janë ADE7753, MSP430AFE2xx, STPM32, MCP39F511A etj.

Qarku i integruar MCP39F511A

MCP39F511A është i dizjanuar për sisteme elektrike një fazore që siguron matje për madhësi të ndryshme elektrike. Një skemë tipike e përdorimit të këtij qarku është treguar në figurën 28.

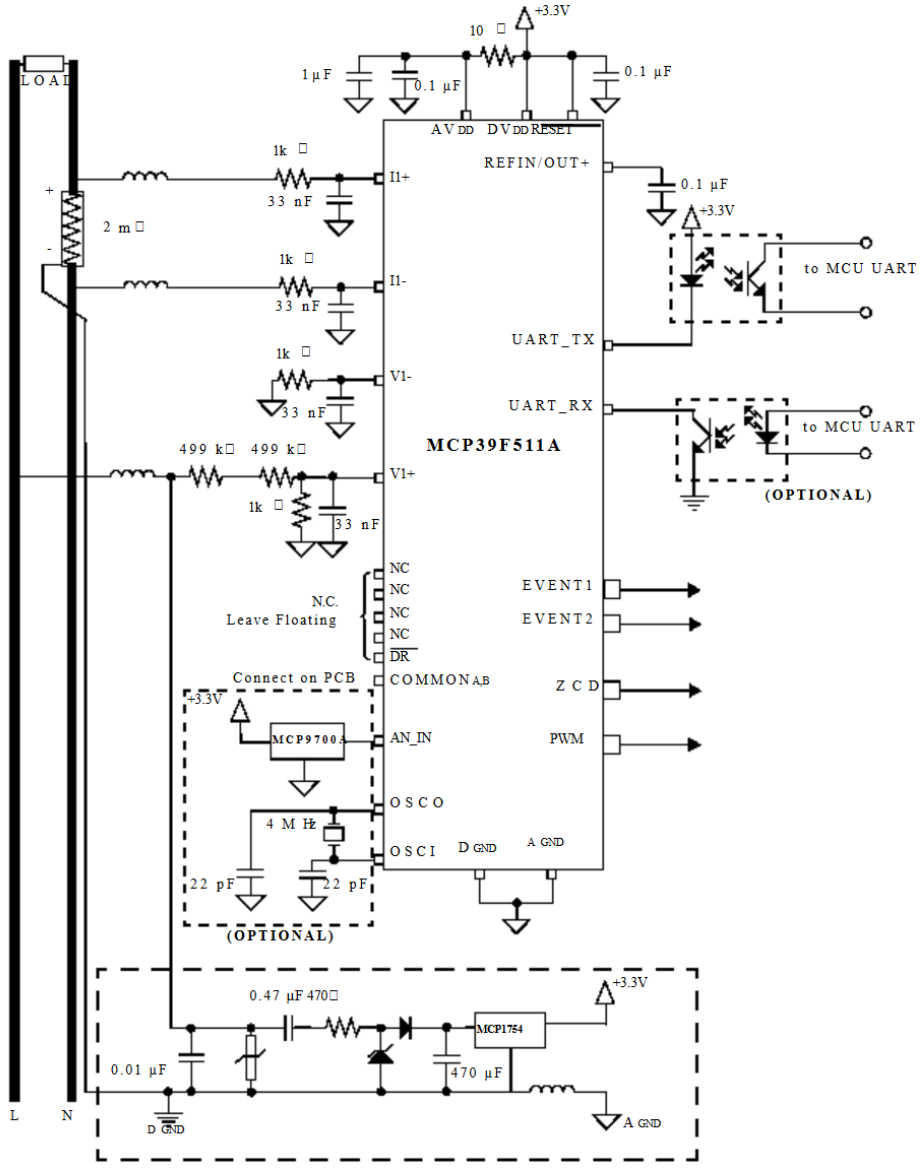


Fig. 28 – Skema e aplikimit të MCP39F511A [13]

Siguron matje në kohë reale për burime të sinjaleve AC dhe DC dhe poashtu ka të implementuar mekanizmin për detektim automatik të ndërrimit të sinjalit në hyrje mes AC dhe DC. Ofron saktësi të lartë me gabim 0.1%. Ka të integruar mikroprocesor 16-bitësh për llogaritje të fuqisë active, reaktive, të dukshme, akumulimit të energjisë, vlerës efektive të rrymës dhe tensionit, frekuencës dhe sinjalit dhe faktorit të fuqisë.

Komponentët e jashtme që mundësojnë matjen për konfiguracionin në figur figurën më lartë janë, një rezistor 2 [mΩ] i lidhur në seri me ngarkesën dhe përdoret për matjen e rrymës, dy rezistor 499 [kΩ] dhe një 1 [kΩ] për ndarës të tensionit me raport 1000:1.

Qarku poashtu ka një pin të dedikuar për PWM, sinjali i të cilit mund të ndryshohet me modifikimin e vlerave të regjistrave përkatws. Pinat Event përdoren për njoftim të ndryshimit të kushteve të brendshme p.sh. në rastin e tensionit shumë të lartë në hyrje. Pini AN_IN është pin hyrës për një shndërrues analog-digjital dhe mund të përdoret për kompenzim kur qarku operon në temperatura të skajshme, sugjerohet të lidhet me termistorin MCP9700A. Pini ZDC (Zero Crossing Detection) gjeneron një sinjal puls i cili është koherent me kalimin e vlerës zero nga sinjali i tensionit në hyrje. Ky sinjal i gjeneruar mund të shërbej për përcaktimin e frekuencës se sinjalit në hyrje nga pajisje të jashtme.

Protokoli i komunikimit për MCP39F511A është i bazuar në Simple Sensor Interface (SSI) protokol. Ky protokol përdoret për komunikime point-to-point mes një mikrokontrolleri dhe një slave të vetëm MCP39F511A. Cdo komunikim pajisjes realizohet në frames të cilët i korrospodojnë formatit më poshtë.

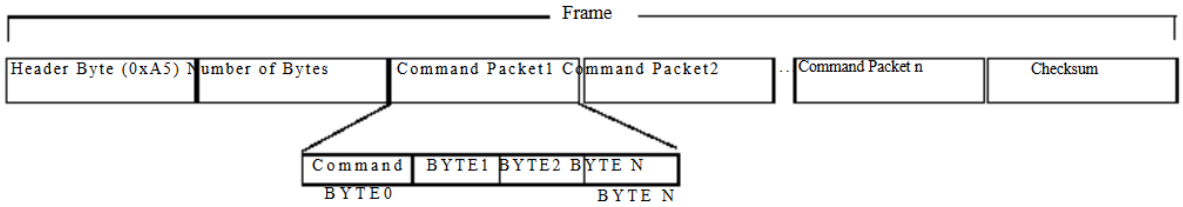


Fig. 29 – Frame i komunikimit me MCP39F511A

Komandat mund të jenë lexim i regjistrave, shkrim në regjistra, lexim i EEPROM, shkrim në EEPROM, kalibrim i gejnit të stadeve hyrëse për tension dhe rrymë, ruatje vlerave të regjistrave në EEPROM etj.

Përfundim

Ky punim ka paraqitur dizajnimin dhe zhvillimin e një pajisje për matjen, shfaqjen dhe regjistrimin e tensionit, rrymës, fuqisë dhe energjisë për DC, duke përdorur një mikrokontrollerin 8051 dhe komponentë të tjerë periferik. Sistemi ofron një zgjidhje efikase për monitorimin e konsumit të energjisë, duke përdorur një kombinim të qarqeve analoge dhe digjitale, për të siguruar matje të sakta dhe të besueshme. Pajisja mund të matë tension, rrymë, fuqi dhe energji, të shfaqë të dhënat në kohë reale në LCD dhe t'i ruajë të dhënat në memorie EEPROM. Kjo tregon potencialin e integrimit të elektronikës analoge dhe digjitale, që mund të përdoren në sisteme automatike dhe teknologji të energjisë efikase. Megjithatë, qarqet me si MCP39F511A janë më të përshtatshme për aplikime të cilat kërkojnë matje të sakta dhe të shpejta me konsum të ulët të energjisë dhe dizajn të thjeshtë, ndërsa përdorimi i mikrokontrollerëve është i dobishëm për sisteme më komplekse dhe që kërkojnë fleksibilitet dhe mundësi për përpunim të avancuar të të dhënave.

Referencat

[1] Neamen, Donald A. *Microelectronics: Circuit Analysis and Design.* 4th ed. Neë York: McGraw-Hill Education, 2012

[2] MacKenzie, Scott. *The 8051 Microcontroller and Embedded Systems: Using Assembly and C*. 2nd ed. Upper Saddle River, NJ: Pearson Education, 2005

[3] Kurtaj, Lavdim. *WUS Script - Mechatronical Project*, Prishtinë: Hasan Prishtina, 2011

[4] Dhaker, Piyu. *Introduction to SPI Interface*. *Analog Dialogue*, 2018. Archived from the original on May 25, 2023. Retrieved July 21, 2023

[5] NXP. *I2C-bus Specification and User Manual*, rev. 6, April 4, 2014

[6] Texas Instruments. *TL072 Operational Amplifier Datasheet*. Last modified [2005]. Available at: [https://www.alldatasheet.com/html-pdf/28775/TI/TL072/19/1/TL072.html].

[7] Atmel (now part of Microchip Technology). *AT89C51 Microcontroller Datasheet*. Available at: [https://www.alldatasheet.com/htmlpdf/56215/ATMEL/AT89C51/126/1/AT89C51.html]

[8] Microchip Technology. *MCP3208 8-Channel, 12-Bit Analog-to-Digital Converter (ADC) Datasheet*. Last modified [2008]. Available at: [https://www.alldatasheet.com/html-pdf/304550/MICROCHIP/MCP3208/486/1/MCP3208.html]

[9] Hitachi. *LM044L LCD Module Datasheet*. Available at: [https://datasheet4u.com/datasheet-pdf/Hitachi/LM044L/pdf.php?id=1462373].

[10] Fairchild Semiconductor (now part of ON Semiconductor). *FM24C256 256Kb Ferroelectric RAM (FRAM) Datasheet*. Last modified [2000]. Available at: [https://www.alldatasheet.com/html-pdf/51902/FAIRCHILD/FM24C256/407/1/FM24C256.html].

[11] Intersil (now part of Monolithic Power Systems). *ICL7107 Precision Analog-to-Digital Converter Datasheet*. Last modified [2005]. Available at: [https://www.alldatasheet.com/html-pdf/67441/INTERSIL/ICL7107/321/1/ICL7107.html].

[12] Schneider Electronics. IEM3250 Energy Meter Datasheet. Available at: [chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://iportal2.schneider-electric.com/Contents/docs/SQD-A9MEM3250_DATASHEET.PDF]

[13] Michrocip Technology. MCP39F511A Energy Meter Datasheet. Available at: [https://www.alldatasheet.com/html-pdf/1284473/MICROCHIP/MCP39F511A/569/1/MCP39F511A.html]

Shtojca A: Leximi i përmbajtjes së EEPROM

```
#include <iostream>
#include <vector>
#include <cstdint>
#include <cstring>
using namespace std;

// Function to convert bytes to float
float bytesToFloat(const uint8_t* bytes) {
    float result;
    // Interpret the bytes as a float
    memcpy(&result, bytes, sizeof(float));
    return result;
}

int main() {
    // Example bytes array representing Voltage, Current, Power, and Energy
     std::vector<uint8_t> bytes = {
        0x00, 0x60, 0x1F, 0x43,
        0xFF, 0x7F, 0x00, 0x3F,
        0x5F, 0xFF, 0x9F, 0x42,
        0x06, 0x88, 0x88, 0x3D,
        0x00, 0x60, 0x1F, 0x43,
        0xFF, 0x2F, 0x36, 0x41,
        0x43, 0xD8, 0xE2, 0x44,
        0x52, 0x22, 0x29, 0x40
    };
```

```
    int i = 0;
    while(4*i < bytes.size()){
        cout<<"Voltage: " << bytesToFloat(&bytes[4*i])<<"  V"<<endl;
        i+=1;
        cout<<"Current: " << bytesToFloat(&bytes[4*i])<<" A"<<endl;
        i+=1;
        cout<<"Power: " << bytesToFloat(&bytes[4*i])<<"    W"<<endl;
        i+=1;
        cout<<"Energy: " << bytesToFloat(&bytes[4*i])<<" Wh"<<endl;
        i+=1;
        cout<<endl;
    }

    return 0;
}
```

Shtojca B: Kodi për AT89C51

Header fajlli adc_mcp3208.h:

```
/*
 *  This delay is needed for MCU and
 *  ADC to be frequency-wise compatible
 */
void adc_delay(void);


/*
 *  SPI communication between ADC and MCU
 *  @param[out] dout
 *
 *  returns digital output generated from ADC
 */
unsigned int adc_spi(__bit channel);
```

Fajlli adc_mcp3208.c:

```
#include "adc_mcp3208.h"
#include <8051.h>

#define CLK P1_0
#define DIN P1_1
#define DOUT P1_2
#define CS P1_3

void adc_delay(){
    return;
}

unsigned int adc_spi(__bit channel) {
    unsigned int dout = 0x0000;
    CS = 1;
    CLK = 0;    //CLK idles low
    DIN = 0;
    CS = 0;
    DIN = 1;
    CLK = 1;    //START = 1
    adc_delay();
    CLK = 0;
    adc_delay();
    CLK = 1;    //SGL = 1
    adc_delay();
    CLK = 0;
    adc_delay();
    DIN = 0;
```

```
    CLK = 1;    //D2 = 0
    adc_delay();
    CLK = 0;
    adc_delay();
    CLK = 1;    //D1 = 0
    adc_delay();
    CLK = 0;
    adc_delay();
    DIN = channel;
    CLK = 1;    //D0 = channel (0 for CH0 and 1 for CH1)
    adc_delay();
    CLK = 0;
    adc_delay();
    CLK = 1;    //11th rising edge
    adc_delay();
    CLK = 0;
    //get 12 bits
    DOUT = 1;
    char i;
    for(i=0; i<13; i++){
       CLK = 1;
       adc_delay();
       dout <<= 1;
       dout |= DOUT & 1;
       CLK = 0;
       adc_delay();
    }
    return dout;
}
```

Header fajlli eeprom_24c256.h:

```
#include <8051.h>

#define bool __bit

/*
 *  This delay is needed for MCU and
 *  EEPROM to be frequency-wise compatible
 */
void eeprom_delay();

/*
 *  I2C communication init between EEPROM and MCU
 *  and prepares conditions for the comm
 */
void i2c_init();

/*
 *  I2C communication starts between EEPROM and MCU
 */
void i2c_start();

/*
 *  Gets acknowledge bit from slave to check for errors
 */
bool getAck();

/*
 *  Writes one page on the EEPROM
```

```
 *  @param[in] slave_address That is address of the slave
 *  @param[in] location_address The first address to start writing
 */
void eeprom_write_page(unsigned char slave_address, unsigned int location_address);

/*
 *  Writes one byte on the EEPROM
 *  @param[in] data Byte to write
 */
void eeprom_write_byte(unsigned char data);

/*
 *  Clears the EEPROM content
 */
void clear_eeprom();

/*
 *  I2C communication stops between EEPROM and MCU
 */
void i2c_stop();
```

Fajlli eeprom_24c256.c:

```
#include "eeprom_24c256.h"

#define SCL P2_2
#define SDA P2_3
#define RED_LED P2_6

unsigned int clr_addr = 0x0000;

void eeprom_delay(){
   char i = 4;
   while(i!=0){i--;}

}
void i2c_init(){
   RED_LED = 0;

   SDA = 1;
   eeprom_delay();
   SCL = 1;
}
void i2c_start(){
   SDA = 0;
   eeprom_delay();
   SCL = 0;
   eeprom_delay();

}
```

```
bool getAck(){
      bool ack_bit;
      SDA = 1;
      SCL = 1;
      eeprom_delay();
      ack_bit = SDA;
      SCL = 0;
      eeprom_delay();
      return ack_bit;

}

void eeprom_write_page(unsigned char slave_address, unsigned int location_address){
   char i = 0;
   for(i = 0; i < 8; i++){
      SDA = (slave_address >> (7-i)) & 1;
      eeprom_delay();
      SCL = 1;
      eeprom_delay();
      SCL = 0;
      eeprom_delay();
   }

   if(getAck() == 1){
      RED_LED = 1;
      i2c_stop();
      return;
   }
```

```
i = 0;
for(i = 0; i < 8; i++){
    SDA = (location_address >> (15-i)) & 1;
    eeprom_delay();
    SCL = 1;
    eeprom_delay();
    SCL = 0;
    eeprom_delay();
}

if(getAck() == 1){
    RED_LED = 1;
    i2c_stop();
    return;
}

i = 0;
for(i = 0; i < 8; i++){
    SDA = (location_address >> (7-i)) & 1;
    eeprom_delay();
    SCL = 1;
    eeprom_delay();
    SCL = 0;
    eeprom_delay();
}

if(getAck() == 1){
    RED_LED = 1;
```

```
        i2c_stop();
        return;
    }
}

void eeprom_write_byte(unsigned char data){
    char i;
    for(i = 0; i < 8; i++){
        SDA = (data >> (7-i)) & 1;
        eeprom_delay();
        SCL = 1;
        eeprom_delay();
        SCL = 0;
        eeprom_delay();
    }

    if(getAck() == 1){
        RED_LED = 1;
        i2c_stop();
        return;
    }
}

void eeprom_delay_clr(){
    int i = 2000;
    while(i!=0){i--;}

}
void clear_eeprom(){
```

```
    int j;
    for(j=0; j<10; j++){ //512 for all
        i2c_start();
        eeprom_write_page(0xa0, clr_addr);

        char i;
        for(i=0; i<64; i++){
            eeprom_write_byte(0x00);
        }
        clr_addr += 64;
        i2c_stop();
        eeprom_delay_clr();
    }
}

void i2c_stop(){
    SCL = 0;
    eeprom_delay();
          SDA = 0;
    eeprom_delay();
          SCL = 1;
    eeprom_delay();
          SDA = 1;
    eeprom_delay();

}
```

Header fajlli lcd_lm044l.h:

```
#define ROW0 0x80
#define ROW1 0xc0
#define ROW2 0x94
#define ROW3 0xd4

/*
 *  This delay is needed for MCU and
 *  LCD to be frequency-wise compatible
 */
void lcd_delay(void);

/*
 *  Used to write commands on LCD
 *  @param[in] command to send on LCD
 */
void lcd_write_cmd(unsigned char command);

/*
 *  Used to write data on LCD
 *  @param[in] data to be written on LCD
 */
void lcd_write_data(unsigned char data);

/*
 *  Used to write data on LCD
 *  @param[in] word Data to be written on LCD
 *  @param[in] size Number of bytes to write
 *  @param[in] row Row of LCD to print on
```

```
 */
void lcd_write_word(unsigned char word[], unsigned char size, unsigned char row);

/*
 * Used to initialize the LCD
 */
void lcd_init(void);

/*
 *  Used to have a delay after
 *  LCD is cleared and because of its
 *  inactivity for that time
 */
void lcd_delay_clr(void);

/*
 * Used to clear the content of LCD
 */
void lcd_clr(void);
```

Fajlli lcd_lm044l.c:

```
#include "lcd_lm044l.h"
#include <8051.h>

#define RS P2_0
#define EN P2_1
#define BUS P0

void lcd_delay(){
    char i = 40;
    while(i!=0){i--;}

}

void lcd_write_cmd(unsigned char command){
    RS = 0;
    lcd_delay();
    EN = 1;
    BUS = command;
    EN = 0;
    lcd_delay();

}

void lcd_write_data(unsigned char data){
    RS = 1;
    lcd_delay();
    EN = 1;
    BUS = data;
```

```
    EN = 0;
    lcd_delay();

}

void lcd_write_word(unsigned char word[], unsigned char size, unsigned char row){
    lcd_write_cmd(row);
    unsigned char i;
    for(i=0; i<size; i++){
        lcd_write_data(word[i]);
    }
}

void lcd_init(){
    lcd_write_cmd(0x38);
    lcd_write_cmd(0x0c);
    lcd_write_cmd(0x80);

    lcd_write_word("Voltage:", 8, ROW0); //0x88
    lcd_write_word("Current:", 8, ROW1);
    lcd_write_word("Power:", 6, ROW2);
    lcd_write_word("Energy:", 7, ROW3);

    lcd_write_word("V", 1, 0x92);
    lcd_write_word("A", 1, 0xd2);
    lcd_write_word("W", 1, 0xa6);
    lcd_write_word("Wh", 2, 0xe6);
}
```

```
void lcd_delay_clr(){
    int i = 3000;
    while(i!=0){i--;}

}

void lcd_clr(){
    lcd_write_cmd(0x01);
    lcd_delay_clr();

}
```

Header fajlli topology_manager.h:

```
#include <8051.h>
#include "adc_mcp3208.h"
#include "lcd_lm044l.h"
#include "eeprom_24c256.h"

/*
 *  Formats voltage and prints it on LCD
 *
 */
void format_print_voltage();

/*
 *  Measures voltage using ADC (CH0)
 *
 */
void measure_voltage();

/*
 *  Formats current and prints it on LCD
 *
 */
void format_print_current();

/*
 *  Measures voltage using ADC (CH1)
 *
 */
```

```
void measure_current();

/*
 *  Formats power and prints it on LCD
 *
 */
void format_print_power();

/*
 *  Calculates the power multiplying actual voltage
 *  and current
 *
 */
void calculate_power();

/*
 *  Formats energy and prints it on LCD
 *
 */
void format_print_energy();

/*
 *  Acumulates energy every sample, adding the power
 *
 */
void calculate_energy();

/*
 *  Increments a seconds counter
```

```
 *
 */
void inc_seconds();

/*
 * Prints the seconds on LCD
 *
 * @param[in] seconds Seconds that have passed
 */
void lcd_print_seconds(unsigned long int seconds);

/*
 *  Stores data on EEPROM
 *
 */
void store_data_eeprom();
```

Fajlli topology_manager.c:

```
#include "topology_manager.h"

#define CHANNEL_0 0
#define CHANNEL_1 1

__code unsigned char lcd_numbers[] = {'0','1','2','3','4','5','6','7','8','9'};

float voltage_volt_f;
float current_amp_f;
float power_wat_f;
float energy_wath_f=0.0;
float printed_energy_f;
__bit is_neg_voltage;
__bit is_neg_current;

unsigned int range_normalize;
unsigned char buffer[7];
unsigned int location_address = 0x0000;

unsigned long int seconds = 0;

void format_print_voltage(){
    buffer[0]='0';
    buffer[1]='0';
    buffer[2]='0';
    buffer[3]='0';
    buffer[4]='0';
```

```
    buffer[5]='0';
    buffer[6]='0';

    if(is_neg_voltage){
        buffer[0] = '-';
    }else{
        buffer[0] = '+';
    }

    int whole_part = (int)voltage_volt_f;
    int fractional_part = (voltage_volt_f - whole_part)*1000;

    if ((fractional_part/10) != 99 && (fractional_part)%10 >= 5){
        fractional_part += 10;
    }

    buffer[1] = lcd_numbers[(whole_part/100)%10];
    buffer[2] = lcd_numbers[(whole_part/10)%10];
    buffer[3] = lcd_numbers[(whole_part)%10];
    buffer[4] = '.';
    buffer[5] = lcd_numbers[(fractional_part/100)%10];
    buffer[6] = lcd_numbers[(fractional_part/10)%10];

    lcd_write_word(buffer, 7, 0x88);
}

void measure_voltage(){

    lcd_delay_clr();
```

```
    unsigned int voltage_volt = adc_spi(CHANNEL_0);

    if(voltage_volt >= 2048){
        is_neg_voltage = 0;
        voltage_volt_f = (((float)voltage_volt/1000.0) - 2.048)*97.65625;
    }else{
        is_neg_voltage = 1;
        voltage_volt_f = (((((float)voltage_volt)/1000.0) - 2.048)*(-1))*97.65625;
    }

    format_print_voltage();
}

void format_print_current(){
    buffer[0]='0';
    buffer[1]='0';
    buffer[2]='0';
    buffer[3]='0';
    buffer[4]='0';
    buffer[5]='0';
    buffer[6]='0';

    if(is_neg_current){
        buffer[0] = '-';
    }else{
        buffer[0] = '+';
    }

    int whole_part = (int)current_amp_f;
```

```
    int fractional_part = (current_amp_f - whole_part)*1000;

    if ((fractional_part/10) != 99 && (fractional_part)%10 >= 5){
        fractional_part += 10;
    }

    buffer[1] = lcd_numbers[(whole_part/100)%10];
    buffer[2] = lcd_numbers[(whole_part/10)%10];
    buffer[3] = lcd_numbers[(whole_part)%10];
    buffer[4] = '.';
    buffer[5] = lcd_numbers[(fractional_part/100)%10];
    buffer[6] = lcd_numbers[(fractional_part/10)%10];

    lcd_write_word(buffer, 7, 0xc8);

}

void measure_current(){
    lcd_delay_clr();
    unsigned int current_amp = adc_spi(CHANNEL_1);

    if(current_amp >= 2048){
        is_neg_current = 0;
        current_amp_f = (((float)current_amp/1000.0) - 2.048)*100.0;
    }else{
        is_neg_current = 1;
        current_amp_f = (((((float)current_amp)/1000.0) - 2.048)*(-1.0))*100.0;
    }
```

```
    format_print_current();

}

void format_print_power(){
    buffer[0]='0';
    buffer[1]='0';
    buffer[2]='0';
    buffer[3]='0';
    buffer[4]='0';
    buffer[5]='0';
    buffer[6]='0';

    int whole_part = (int)power_wat_f;
    int fractional_part = (power_wat_f - whole_part)*100;

    buffer[0] = lcd_numbers[(whole_part/1000)%10];
    buffer[1] = lcd_numbers[(whole_part/100)%10];
    buffer[2] = lcd_numbers[(whole_part/10)%10];
    buffer[3] = lcd_numbers[(whole_part)%10];
    buffer[4] = '.';
    buffer[5] = lcd_numbers[(fractional_part/10)%10];
    buffer[6] = lcd_numbers[(fractional_part)%10];

    lcd_write_word(buffer, 7, 0x9a);

}

void calculate_power(){
```

```
    power_wat_f = voltage_volt_f * current_amp_f;

    format_print_power();
}

void lcd_print_seconds(unsigned long int seconds){
    unsigned char q = seconds/10;
    unsigned char dh = seconds%10;

    buffer[0] = lcd_numbers[q];
    buffer[1] = lcd_numbers[dh];

    lcd_write_word(buffer, 2, 0xa3);
}

void format_print_energy(){
    buffer[0]='0';
    buffer[1]='0';
    buffer[2]='0';
    buffer[3]='0';
    buffer[4]='0';
    buffer[5]='0';
    buffer[6]='0';

    int whole_part = (int)energy_wath_f;
    int fractional_part = (energy_wath_f - whole_part)*100;

    buffer[0] = lcd_numbers[(whole_part/1000)%10];
    buffer[1] = lcd_numbers[(whole_part/100)%10];
```

```
    buffer[2] = lcd_numbers[(whole_part/10)%10];
    buffer[3] = lcd_numbers[(whole_part)%10];
    buffer[4] = '.';
    buffer[5] = lcd_numbers[(fractional_part/10)%10];
    buffer[6] = lcd_numbers[(fractional_part)%10];

    lcd_write_word(buffer, 7, 0xdb);

}

void calculate_energy(){
    energy_wath_f = energy_wath_f + (power_wat_f*(1.0/3600.0)); // integrate power
    //lcd_print_seconds(seconds);
    format_print_energy();

}

void store_data_eeprom(){
    i2c_start();
    eeprom_write_page(0xa0, location_address);

    unsigned char* byte_f =  (unsigned char*)&voltage_volt_f;
    eeprom_write_byte(byte_f[0]);
    eeprom_write_byte(byte_f[1]);
    eeprom_write_byte(byte_f[2]);
    eeprom_write_byte(byte_f[3]);

    byte_f =  (unsigned char*)&current_amp_f;
    eeprom_write_byte(byte_f[0]);
```

```
    eeprom_write_byte(byte_f[1]);
    eeprom_write_byte(byte_f[2]);
    eeprom_write_byte(byte_f[3]);

    byte_f =  (unsigned char*)&power_wat_f;
    eeprom_write_byte(byte_f[0]);
    eeprom_write_byte(byte_f[1]);
    eeprom_write_byte(byte_f[2]);
    eeprom_write_byte(byte_f[3]);

    byte_f =  (unsigned char*)&energy_wath_f;
    eeprom_write_byte(byte_f[0]);
    eeprom_write_byte(byte_f[1]);
    eeprom_write_byte(byte_f[2]);
    eeprom_write_byte(byte_f[3]);

    location_address += 16;
    i2c_stop();
}

void inc_seconds(){
    seconds++;
}
```

Fajlli main.c:

```
/* Main.c file generated by New Project wizard
 *
 * Created:   Wed Jan 31 2024
 * Processor: AT89C51
 * Compiler:  SDCC for 8051
 */

#include <8051.h>
#include "topology_manager.h"

#define SAVE_DATA P2_4
__bit SAMPLE_DATA;
__bit STORE_DATA;

unsigned char quarter_ms = 0;
unsigned int ms = 0;

void T0_ISR(void) __interrupt(1){
   quarter_ms++;
   if (quarter_ms == 4){
      quarter_ms = 0;
      ms++;
      //SAMPLE_DATA = 1;

      if (ms == 1000){
         inc_seconds();
         SAMPLE_DATA = 1;
         ms = 0;
```

```
        }

        //SAVE_DATA = 1;
        if (!SAVE_DATA){
            while (!SAVE_DATA){}
            STORE_DATA = 1;
        }

    }

}

void t0isr_init(){
    TMOD = 0x02;
    IE = 0x82;
    TL0 = 6;
    TH0 = 6;
    TR0 = 1;

}

void main(void)
 {
    t0isr_init();
    lcd_init();
    i2c_init();
    SAMPLE_DATA = 1;
    //clear_eeprom();
```

```
  while (1){
    if (SAMPLE_DATA){
      measure_voltage();
      measure_current();
      calculate_power();
      calculate_energy();
      SAMPLE_DATA = 0;
    }

    if (STORE_DATA){
      store_data_eeprom();
      STORE_DATA = 0;
    }
  }
}
```