KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Jurgis Vaičiūnas

REALAUS LAIKO APLINKOS MONITORINGO SISTEMOS SUDARYMAS IR TYRIMAS

Magistro darbas

Darbo vadovas

prof. dr. Egidijus Kazanavičius

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Jurgis Vaičiūnas

REALAUS LAIKO APLINKOS MONITORINGO SISTEMOS SUDARYMAS IR TYRIMAS

Magistro darbas

Recenzentas

doc. A. Venčkauskas

2011-05-

Vadovas

prof. dr. E. Kazanavičius

2011-05-

Atliko

IFM-9/1 gr. stud.

Jurgis Vaičiūnas

2011-05-27

Kaunas, 2011

ĮVADAS

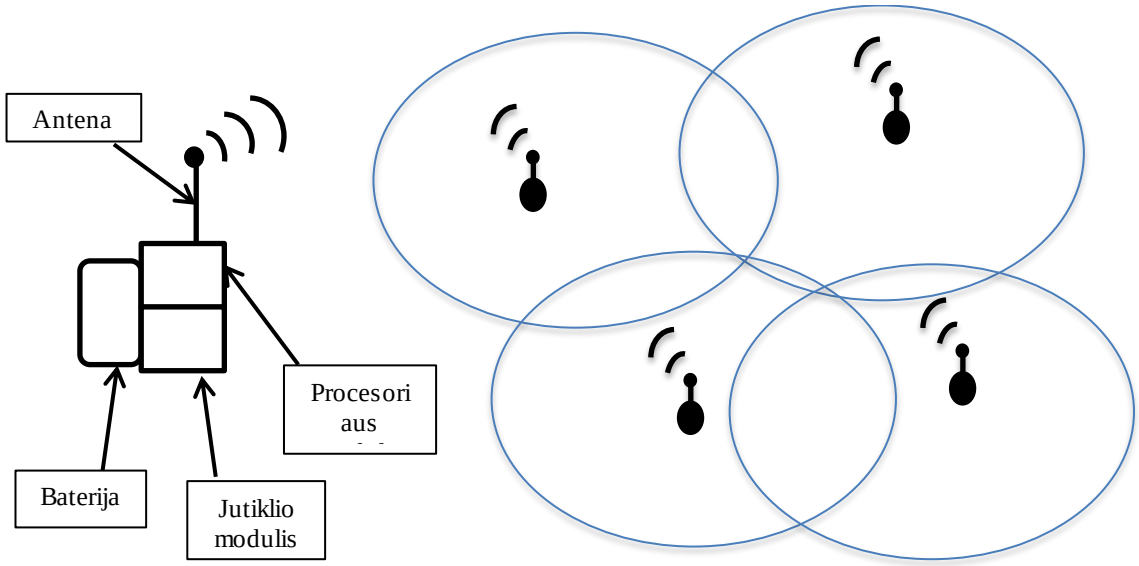
Informacija yra vienas vertingiausių dalykų. Ją kaupiame apie įvairiose srityse. Tik turint tinkamą informaciją galima tinkamai reaguoti į įvykius ir priimti tinkamus sprendimus.

Problema yra tame, kad nėra pigaus, efektyvaus ir greitai parengiamo būdo aplinkai stebėti. Aplinkos stebėjimui kartais pasirenkama speciali stotis su interneto jungtimi, tačiau šis variantas nėra tinkamas jei duomenis norima rinkti tam tikrame plote, nes tada reikėtų vedžioti kabelius iki kiekvienos stotelės. Kuo daugiau kabelių tuo didesnė rizika, kad jie bus pažeisti. Be to norint pridėti naujų stebėjimo parametrų reikia arba modifikuoti stoteles ir vedžioti naujus kabelius. Taigi efektyvaus ir lankstaus būdo aplinkos stebėjimui nėra.

Dar vienas būdas aplinkai stebėti yra pasinaudojant žmogiškaisiais resursais, tačiau, tai tampa nepraktiška, jei stebėjimas trunka ilgai, nes kaina lyginant su automatizuotom sistemom išauga. Jei reikia stebėti keliose vietose, tokiu atveju priklausomai nuo stebimo ploto ar atstumo tarp stebimų vietų reikia transporto priemonės. Jei stebėjimas vyksta visą parą didėja tikimybė, kad dėl žmogiškosios klaidos, kažkoks įvykis liks nepastebėtas.

Darbo objektas ir būdas šiai problemai spręsti galėtų būti dabar labai populiarėjantys bevieliai jutiklių tinklai. Kiekvienas toks jutiklis susideda iš baterijos, bevielio ryšio antenos ir procesoriaus, bei jutiklio modulio, kuris parenkamas priklausomai nuo tinklo paskirties. Tokie jutikliai paskirstomi stebimoje aplinkoje ir prireikus gali būti perkelti į kitą vietą ar pašalinti. Sukaupta informacija laikoma pačiuose jutikliuose, todėl nebūtina ištisai tikrinti jų turimą informaciją, ją galima surinkti ir vėliau. Taip pat iki kiekvieno daviklio nereikia vesti laidų, nes energiją jie gauna iš baterijų, o informacija perduodama bevieliu būdu. Pašalinant iš sistemos laidus padidėja jos patikimumas, nes nutraukus laidą prarandamas ryšys su sistema. Taip pat duomenis surinkti, užtektų pastatyti tarpinius daviklius ir informacija galėtų būti surenkama per juos, taigi pašalinama būtinybė fiziškai keliauti iki stebimos aplinkos. Pradėti stebėti naujus parametrus su tokia sistema taip pat būtų gana paprasta, nes užtektų padėti naują daviklį ir jis tiesiog pradėtų bendrauti su jau esančiais.

1 pav.Tinklo sudėtis



Stebėjimo objektu šiame darbe pasirinksime pastatą, pavasarį esantį potvynio rizikos zonoje, o vasaros pabaigoje gaisrų pavojaus zonoje.

Darbo tikslas yra išanalizuoti įvairius bevielius informacijos perdavimo būdus, jutiklių tipus ir tinkamumą, programinę įrangą tinkamą naudoti šio tipo sistemose, energijos panaudojimo efektyvumą bei duomenų surinkimo ir saugojimo būdus ir sudaryti tokio tipo sistemos modelį.

Darbo uždaviniai

- Atlikti jutiklių ir jų parametrų analizę
- Atlikti jutiklių programinės įrangos analizę
- Atlikti duomenų perdavimo būdų efektyvumo analizę
- Atlikti energijos valdymo tinkle analizę
- Sudaryti šio tinklo struktūrą, bei veikimo algoritmą
- Sudaryti jutiklių informacijos apdorojimo būdą
- Parinkti duomenų kaupimo ir perdavimo būdą šiai sistemai
- Atlikti sistemos ilgaamžiškumo tyrimą

1. BEVIELIŲ JUTIKLIŲ TINKLŲ ANALIZĖ

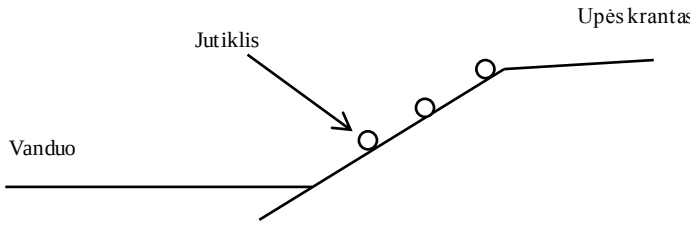
1.1 Aplinkos stebėjimo jutiklių analizė

Kokie jutikliai bus naudojami daugiausiai priklauso nuo stebėjimų srities. Tačiau jie taip pat turėtų būti nebrangūs, kad nekiltų didelė pagunda juos vogti, taip pat turėtų būti nedideli, bei vartoti nedaug energijos. Taip pat reikia įvertinti kokią informaciją norima rinkti, nes nuo to priklauso paties mazgo parametrai.

Vandens jutikliai būtų naudojami vietovėse su potvynio rizika, netoli upių ar kitų vandens šaltinių.

Panaudojimas. Būtų išdėliojami keliomis eilėmis, kad būtų žinomas lygis, kurį pasiekė vanduo.

2 pav. Vandens jutiklių išdėstymas



Tokio tipo jutikliai būtų labai būtų pravertę, kai Nėris ir Nemunas patvino, žmonės atsibudo aplieti vandens, jei palei upės krantą būtų padėti tokie davikliai (matuojantys kokią ribą pasiekė vanduo), daug turto būtų buvę galima išgelbėti. Suveikus tokiam davikliui signalas galėtų automatiškai būti perduodamas susijusioms tarnyboms.

Stebint potvynius, aliarmą reikėtų skelbti tik tada kai bent keli tame pačiame lygyje esantys jutikliai duotų signalą, nes jei apsemiamas tik vienas jutiklis, tai gali būti tik trumpas užpylimas arba klaida.

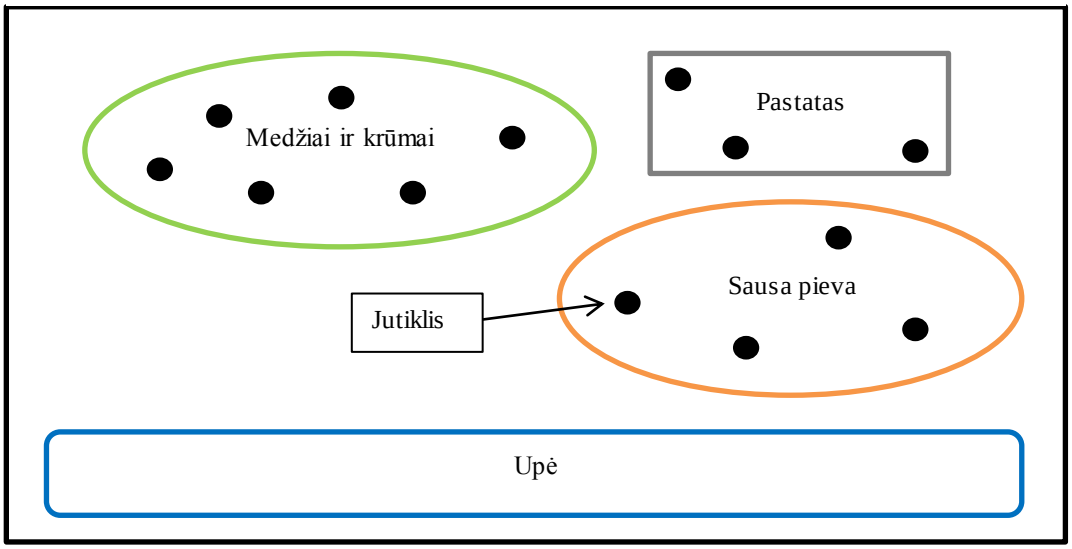
Pagrindiniai parametrai:

- Atsparumas drėgmei ir vandeniui
- Atsparumas korozijai
- Galimybė įtvirtinti

Ugnies jutikliai būtų naudojami ten, kur yra gaisro rizika. Šie jutikliai yra universalūs todėl gali būti naudojami ir pastatuose ir lauko sąlygomis.

Panaudojimas. Galima pažymėti kokioje zonoje yra jutiklis ir konkrečiam jutikliui davus signalą, būtų žinoma gaisro vieta. Šiame paveikslėlyje pavaizduotos trys zonos: pastato, medžių ir krūmų, bei sausos pievos.

3 pav. Ugnies jutiklių išdėstymas



Pagrindinai parametrai:

- Atsparumas aukštoms temperatūroms
- Atsparumas drėgmei ir vandeniu (ypač aktualu mazgams esantiems potvynio rizikos zonoje)
- Atsparumas korozijai

Temperatūros jutikliai galėtų būti montuojami ten kur svarbu temperatūrą išlaikyti tolygią, arba, kad neviršytų nustatytos ribos: sandėliuose, fabrikuose , kur gamybos proceso metu tai labai svarbu, pavyzdžiui didesniuose šaldytuvuose.

Panaudojimas. Šie jutikliai būtų paprasčiausiai išdėliojami tokiais atstumais, kokiais aktualu žinoti temperatūrą.

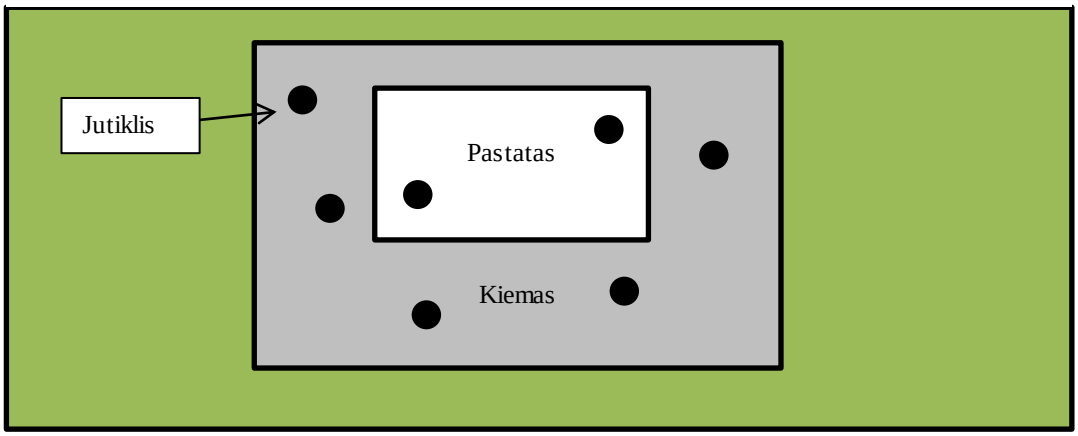
Pagrindinai parametrai:

Kadangi šie mazgai bus pastato viduje ir paprasčiausiai matuos temperatūrą, jokių išskirtinių reikalavimų jiems nėra.

Judesio jutikliai būtų dėstomi ten kur reikalinga apsauga. Būtų galima tiksliai fiksuoti kur juda stebimas objektas ir kiek jų yra.

Panaudojimas. Šie mazgai taip pat, kaip ir ugnies jutikliai būtų išdėliojami zonomis. Taip būtų galima žinoti, kaip giliai pavyko įsibrauti. Šiame paveikslėlyje mazgai suskirstyti į dvi grupes: kiemo ir pastato.

4 pav. Judesio jutiklių išdėstymas



Pagrindinai parametrai:

- Privalumas būtų nustatyti judančio objekto dydį

1.2 Bevielių jutiklių programinės įrangos analizė

Pagrindinės savybės apibūdinančios wsn tašką būtų:

- Ribota galia, kurią jie gali gauti ir laikyti
- Galimybė išsilaikyti sunkiomis aplinkos sąlygomis
- Mobilumas
- Dinaminis tinklo išsidėstymas
- Bendravimo nesklandumai
- Taškų heterogeniškumas
- Didelis panaudojimo mastas
- Funkcionavimas priežiūros
- Taškų kiekio kintamumas

Norint suderinti šias savybes reikalinga, labai mažai resursų vartojanti, tačiau stabili ir lanksti operacinė sistema. Jau yra sukurtos operacinės sistemos skirtos būtent tokio tipo įrenginiams. Pagrindinės būtų: RETOS, LIMOS, TinyOS, Mantis, Contiki, t-kernel.

Retos (Resilient, Extensible, and Threaded OS) kaip ir sako pavadinimas, tai lanksti, dinamiška ir „daugiagijė“ operacinė sistema. Lanksti tai reiškia, kad galima keisti jos parametrus. Dinamiška – esamų jutiklių kiekis sistemoje gali kisti. „Daugiagijė“ – sistema veikia naudojant kelias gijas. Retos turi keturis pagrindinius tikslus, kurie yra suteikti „daugiagijiško“ programavimo sąsają, sistemos tvirtumas, branduolio išplėtimas su dinaminiu perkonfigūravimu, WSN orientuoto tinklo palaikymas.

Retos suteikia programinę galimybę atskirti branduolį nuo vartotojo programų ir palaiko stabilų jų veikimą mažai atminties turinčiuose įrenginiuose. Tai padaroma dvigubo veikimo rėžimu, jis logiškai atskiria branduolio ir vartotojo veikimo vietas. Taip pat programų veikimas yra nuolatos stebimas, kad nesuvartotų per daug atminties ir neužlūžtų dėl resursų trūkumo. Taip pat branduolys gali būti perkonfigūruotas, per užkraunamo branduolio struktūrą, kad būtų galima sukonstruoti branduolį kuris būtų optimizuotas tuo metu veikiančiai programai ir vartotų mažai resursų. Taip pat RETOS yra sukurtas su kelių lygių tinklo architektūra, kuri būtų patogi naudoti WSN sistemose.

Retos taip pat palaiko „daugiagijiškumą“, taigi programuotojams nereikia jaudintis dėl optimalaus jų programų veikimo ir jie gali susikoncentruoti į uždavinio semantiką. Taip yra dėl to kad nereikia galvoti apie procesų konkurenciją, tokią kaip įvykiais valdomoje aplinkoje.

Kad sumažintų atminties naudojamą branduoliui RETOS naudoja dvi technologijas. Vieno branduolio stekas sumažina gijos steko resursų reikalavimus, o steko dydžio analizė padaro, kad kiekvienai gijai automatiškai būtų skiriama duomenų tiek kiek reikia [2].

TinysOS (Tiny Operating System) tai dar viena operacinė sistema skirta bevielių jutiklių tinklams. Ši platforma jau yra gana paplitusi ir naudojama įvairių rūšių bevieliuose tinkluose. TinyOS pasižymi tuo, kad yra modulinio dizaino. Tai reiškia, kad ji gali susidėti iš daugybės skirtingos paskirties dalių, kurios tarpusavyje bendrauja per specialias sąsajas. TinyOS buvo kuriama su tikslu, kad vėliau ji bus diegiama „smartdust“ protingose dulkėse.

David E. Culler (2006) rašo, kad tai yra asinchroninė operacinė sistema ir turi tik vieną bendrą steką. Tai reiškia, kad procesas gali veikti nesulaukęs kito proceso pabaigos. Tai turi privalumų, pavyzdžiui nuskaitymo/įrašymo operacijos yra itin lėtos lyginant su paties procesoriaus darbu, todėl kad vyksta mechaniški procesai, taigi jei tarkim kažkoks procesas laukia kol bus nuskaityti duomenys, jis gali laukti gana ilgai jei tų duomenų bus daug, asinchroninė sistema leis pradėti apdoroti duomenis tik pradėjus juos nuskaityti. Dėl tokios sistemos taip pat apmažėja konkurencija, nes dirbant kuriam nors prietaisui staiga gali prireikti ir kito. Didesnėje sistemoje vienas prietaisas tiesiog nueitų į laukimo rėžimą, kol suveiktų didesnį pirmumą turintis. Tačiau tai tinka tik didesnėms sistemoms kurios turi daugiau resursų. Integruotose sistemose labiau tinkama įvykiais valdoma sistema, ji leidžia labiau reguliuoti sistemos darbo tvarkaraštį, be to reikalauja mažiau resursų. Tačiau bėda yra ta, kad net ir nedideli pakeitimai gali reikalauti didelių žinių.

TinysOS naudoja struktūrizuotą įvykiais valdomą architektūrą. Visa sistema susideda iš daugybės modulių, kurie tarpusavyje sujungiami per sąsajas. Komponentai yra atskirti vieni nuo kitų ir turi gerai apibrėžtas sąsajas, vidines būsenas ir vidinę konkurenciją. Pirminiai struktūros elementai valdo kietąją dalį, jų sąsajos atspindi jos veikimą ir pertraukimus, būseną ir konkurenciją. Aukštesnio lygio komponentai panašiu principu gaubia programinės įrangos funkcionalumą. Jie suteikia informaciją apie komandas, signalus, įvykius, gijas, kintamuosius

ir turi vidinį valdymą. Tai leidžia nesunkiai keisti arba tobulinti techninę dalį. Be to suteikia sistemai lankstumo, atsparumo ir supaprastina programavimą. Kiekviename komponente veikia sava supaprastinta gija, kuri vadinama „task“ užduotis, tačiau bendravimas tarp komponentų vyksta tik per sąsajas. Tokia struktūra supaprastina tvarkaraščių sudarinėjimą. TinyOS 2.0 versijoje tvarkaraščių sudarymo modulis taip pat gali būti pakeistas.

TinyOS parašyta nesC kalba, kuri yra skirta bevielių monitoringų tinklų tobulinimui ir palaiko modulinių komponentų kūrimą, turi visos sistemos analizę atliekančius įrankius, taip pat gali atlikti tinklo duomenų analizę. Taip pat kompiliatorius atlieka visos hierarchinės struktūros patikrinimą, pažiūri ar nėra rakinimo aklaviečių (deadlock), taip pat patikrina ar nevartojama per daug resursų. Taip pat TinyOS bendruomenė yra sukūrusi keletą įskiepių jau sukurtoms integruotoms kūrimo aplinkoms.

Tinklo duomenų tipai supaprastina protokolų implementaciją. Tinklo duomenys turi tam tikrą formatą, tuo tarpu paprasti duomenys priklauso nuo žodžio ilgio ir procesoriaus adresavimo, taigi paprastai reikėtų nemažai laiko sugaištama duomenų pervertimui iš vienų į kitus. Kadangi TinyOS naudoja tinklo duomenų tipus, kurie yra labai aiškiai apibrėžti, kompiliatorius suteikia labai efektyvų priėjimą prie tam tikrų paketo vietų davikliuose ir JAVA bei XML metodus darbui su šiais duomenimis tradiciniuose kompiuteriuose.

TinyOS komponentai yra susieti dvipusiais ryšiais. Kai aukštesnio lygio komponentas iššaukia komandą kažką vykdyti techninėje arba programinėje dalyje, komanda iškarto grįžta ir praneša apie reikalavimo būklę. Nėra svarbu tai kad darbas gali užtrukti ilgiau, kai darbas baigiamas, duodamas signalas ir toliau vykdomi sekantys veiksmai. Kol laukiama signalo valdantysis komponentas gali duoti nurodymus kitiems įrenginiams arba laukti budėjimo rėžime.

TinyOS turi penkias posistemes: jutikliai, ryšio priemonės, laikmenos, laikmačiai ir procesoriaus/energijos suvartojimo valdymo sistema. Atviro kodo saugykla suteikia komponentus populiarioms mikroschemoms įskaitant ir kelis mikrovaldiklius, radijo imtuvus, flash atmintines, analoginio signalo į skaitmeninį keitiklius, sensorius, bei konfigūracijas kelioms populiarioms platformoms. Protingai parinktos sąsajos leidžia projektuotojams pasirinkti ir iš nuo techninės dalies priklausomos ir nepriklausomos įrangos.

Įprastos sistemos būna paruoštos tam tikriems įrenginiams su prietaisų programine įranga. Bevielių jutiklių tinkluose, įvairūs jutikliai gali būti naudojami įprastose sistemose. TinyOS turi įvairių prietaisų valdymo programinės įrangos. Kadangi sensorių pasirinkimas

yra labai susietas su aplikacija ir techniniu surinkimu, prietaisus valdanti programinė įranga turi būti lengvai konfigūruojama ir bandoma, o ne pridedama vėliau.

TinyOS tobulėjo kartu su radijo ir kitomis ryšio priemonėmis. Komunikacijų komponentas gali dirbti realiu laiku, kol kiti komponentai dirba, reaguoti į įvykius. Pradinės versijos bendravo paprastai bitas po bito, vėliau žodžiai ilgėjo, kol galiausiai pasiekė pradėta bendrauti paketų lygio sąsajom. Naujausia TinyOS versija suteikia vienodą sąsają visoms radijo mikroschemos galimybėms. Ryšio lygio komponentas suteikia daug galimybių valdyti prietaisus (MAC) galimybės, įskaitant ir kanalo aktyvumo aptikimą, kolizijų vengimą, duomenų perdavimą, grafikų sudarymą ir energijos valdymą, o tai leidžia komponentus paruošti įvairiems protokolams.

Įvykių registrams, nustatymams, failams ir programoms bevielių jutiklių tinkluose dažniausiai naudojama pastovioji atmintis. Naudojamos ir kelios skirtingos „flash“ atmintinės, kurios turi skirtingas sąsajas ir skirtingus protokolus.

Dažniausiai naudojamos paslaugos bevieliuose jutiklių tinkluose yra žinučių skleidimo užtikrinimas, bendro duomenų saugojimo paslauga ir maršrutų valdymas.

Žinučių skleidimo užtikrinimo paslauga garantuoja žinučių nusiuntimą iš šaltinio, dažniausiai iš prietaiso kuris išeina į kitus tinklus („gateway“), į visus ar kelis mazgus. Jis naudojamas tinklo nustatymams keisti bei valdymui ir nepriklauso nuo to ar tinklas prieš tai atliko gretima esančių mazgų paiešką ar sudarė kokius nors maršrutus ir nuo to kokie nustatymai. Kiekviena užtikrinimo operacija užpilo visą tinklą pranešimais, kurie pakeičia mazgų nustatymus arba duoda su tuo susijusias instrukcijas. Kai vienas mazgas gauna kažkokias instrukcijas, jis jas perduoda gretimiems mazgams. Mazgas gali būti nepasiekiamas, kelias gali būti blokuojamas arba jo baterija gali būti išsekusi, kai jam perduodamos instrukcijos, taigi kartas nuo karto perduodamos instrukcijos perduodamos instrukcijos užtikrina, kad visų mazgų informacija būtų atnaujinta. Jei tinklas būtų labai tankus, mazgai sulaikytų instrukcijų persiuntimus išgirdę iš gretima esančių mazgų ateinančią informaciją. Patikimumas yra pasiekiamas per tinklo mazgų gausą ir nuolatinį duomenų tvarkymą.

Duomenų surinkimui ir tvarkymui bevielių jutiklių tinklai sukuria vieną ir daugiau medžių kurie susijungia ties duomenų surinkimo prietaisu. Kiekvienas mazgas renka ryšio kokybės statistiką ir kelių vertes (kiek energijos suvartojama keliaujant iš vieno mazgo į kitą) visomis ryšio priemonėmis įskaitant paketų persiuntimą ir pasyvų srauto stebėjimą . Mazgas naudoja šiuos duomenis, kad galėtų parinkti geriausią kelią iki aukštesnių medžio šakų tiek

gautų duomenų perdavimui iš žemesnių šakų tiek savųjų siuntimui. Kadangi mazgų skaičius, ryšio nuotoliu gali labai skirtis, parenkamų kaimynų skaičius turi būti fiksuotas bet kuriam tinklui.

Panašiu principu bet kuris mazgas gali parinkti kelią iki bet kurio kito ir persiųsti paketus žingsnis po žingsnio. Toks maršrutų sudarymas dažniausiai naudojamas perduoti komandoms konkrečiam mazgui.

Paprastai TinyOS aplikacija yra maža programėlė sukurta dirbti naudojant sistemos paslaugas. Ji paleidinėja jutiklio valdiklius, taip pat apdoroja gautus duomenis. Taip pat ji gali kaupti apdorotus duomenis. Taip pat ji gali naudoti visas ryšio priemones bendravimui su kitais mazgais. Tiek sistemos paslaugos tiek aplikacijos gali būti valdomos bevieliu būdu per žinučių skleidimo paslaugą.

Dauguma aplikacijų gali būti modifikuotos daugybei funkcijų atlikti. Pavyzdžiui aplinkos stebėjime, duomenų surinkimo dažnis ir aliarmo riba yra natūraliai konfigūruojami parametrai.

Kadangi bevielių jutiklių tinklo technologijos tobulėja ji suteikia galimybes stebėti pasaulį tokiu tikslumu, kokiu anksčiau buvo nepraktiška ar neįmanoma. Patobulėjo puslaidininkiai, mobilumas. Atsirado efektyviai energiją vartojančių radijo prietaisų. Algoritmų kūrimas ir praktiniai bandymai leido sukurti įmantrias aplikacijas [3].

Limos (Lightweight Multi-Threading Operating System) lengva „daugiaugijė“ operacinė sistema skirta bevielių jutiklių tinklams. Limos yra yra gudri, sugeba prisitaikyti prie situacijos, reaguojanti į turimus resursus, kurių stengiasi suvartoti kuo mažiau, taip pat ji turi realiu laiku veikiantį mikro branduolį. Šios sistemos architektūra susideda iš dviejų dalių: iš reagavimo į įvykių dalies. Įvykis yra pačios LIMOS darbinė dalis, o gija yra įvykio dalis. Taip pat dviejų lygių tvarkaraščių sistemą: vieną aukšto lygio gijoms, kitą žemo. Gijos naudoja bendrą atmintį, kuri LIMOS yra realizuota bendru surikiuotu sąrašu (tuple). Ši sistema taip pat atsižvelgia ir į realiu laiku veikiančias aplikacijas.

Kiekvienas įvykis yra atsakingas už kažkokią užduotį, kuri paleidžiama tik ją iššaukiant per („ISR“ interrupt service routine) pertraukimų paslaugą arba būti iššaukiamas kitų įvykių. Limos įvykiai gali būti susieti tik su vienu pertraukimų šaltiniu, bet su keliais išvedimų šaltiniais.

Įvykiai taip pat skirstomi:

- Nuolatinius - jie tinkami duomenų rinkimui per sensorius arba sistemos kontroliavimui. Tokių įvykių darbo laikas yra žinomas.
- Pavienius - kurie naudojami apdoroti kritines neplanuotas užduotis.

  Taip pat įvykiai gali skirstomi:
- Kritinius - tai įvykiai kurie turi būtinai sureaguoti per jiems duotą laiko tarpą. Tokie įvykiai gali būti iššaukiami medicininės įrangos sistemų, taip pat saugos sistemų.
- Paprastus - tai įvykiai kurie turėtų būti įvykdyti per duotą laiką, tačiau nėra gyvybiškai svarbūs. Tai gali būti sistemos daviklių informacijos atnaujinimas ar panašios reikšmės darbas.

Gijos – tai mažesnė įvykių dalis ir jos bendrauja tik tarpusavyje su to paties įvykio gijomis per bendrą atmintį. Jos dirba lygiagrečiai. Kadangi gijos naudojasi tais pačiais įvykio resursais, tai sumažina gijų junginėjimąsi tarpusavyje ir sutaupo energijos.

Architektūra įvykis/gija iš pirmo žvilgsnio atrodo tokia pati kaip ir kitų operacinių sistemų procesas/gija, kur procesas tapatinamas su įvykiu, tačiau dėl skirtingos paskirties šios dvi architektūros skiriasi. Įvykis yra neplanuotas ir dažniausiai labai aukšto prioriteto. Ir dėl to ši architektūra labiau tinkama bevielių jutiklių sistemoms.

Sistemos darbo planavimas yra viena pagrindinių sistemos dalių, todėl, kad yra labai svarbu žinoti, kas po ko veiks. Limos turi dviejų lygių planavimą:

- skirtą paprastiems įvykiams.
- Skirtą kritiniams įvykiams.

Limos įvykių tvarkaraštyje nenaudoja išankstinio planavimo, visi I/O prietaisai veikia pertraukimų pagalba ir tada priklausomai nuo prioritetų jie iššaukiami. Iššaukus įvykį jis be jokio pasiruošimo veikia iki galo. Kai vienas įvykis baigia darbą, sistema iškart parenka kitą įvykį. Tai dinaminių prioritetų planavimas, kuris veikia kaip EDF (earliest deadline first) artimiausias galutinis terminas pirma. Tas įvykis kurio galutinis terminas pasibaigs greičiausiai paleidžiamas pirmas. Toks planavimas padaro tvarkaraštį nuoseklų bei lankstų. Įvykiai tvarkaraštyje bėgimo metu gali būti pertraukiami kito, su aukštesniu prioritetu įvykio. Kai įvyksta pertraukimas ISR pertraukimų paslauga nukelia jį į gretimą atminties sąrašą. Tolimesni veiksmai su tuo įvykiu yra atliekami tik tada, kai kitas įvykis baigia savo darbą.

Periodiniam įvykiui sistema duoda galutinį terminą lygų jo periodui Di = pi , tuo tarpu, kai įvyksta netikėtas įvykis, jam duodamas maksimalus galimas laikas Di .

Gijų tvarkaraščiui Limos naudoja prioritetais paremtą iš anksto paruoštą sistemą. Gijos pradeda veikti priklausomai nuo prioriteto ir aukštesnio lygo gija gali paruošti darbui kitą, žemesnio lygio giją, išskyrus tuos atvejus jei įvyksta kritinis pertraukimas.

Kai vieno įvykio gijos veikia, tai kito įvykio gijos negali prieiti prie procesoriaus resursų, tai leidžia įvykiams veikti iki galo.

Gijų planavimo schema yra statinė ir planuoja pagal prioritetus, tai reiškia, kad tvarkaraštis sudaromas ir fiksuojamas pagal sąsajas tarp gijų dar prieš paleidžiant. Prioritetai tarp gijų turi būti parenkami atsargiai, kad nepakliūti į rakinimo aklavietę (deadlock). Taip nutinka kai du ar daugiau procesų negali veikti nes laukia vienas kito, kad pasibaigtų. Pavyzdžiui jei aukšto prioriteto gija yra pertraukiama ir laukia duomenų iš vieno atminties sąrašo, o kita žemesnio lygio gija veikia savo atminties sąraše ir vienintelė gali suteikti duomenis aukšto prioriteto gijai, tačiau laukia kol jis pasibaigs. Taigi, kad išvengti tokių situacijų Limos prioritetus skirsto pagal šią schemą:

1) Tarkim, kad gija laukia signalo iš įvykio εi ir eina po gijos τi0 kuris priklauso εi , tokiu atveju gija τi0 turi aukščiausią prioritetą.
2) Tariama, kad kiekviena gija priklauso tik vienam atminties sąrašui (tuple)
3) Todėl gija τi,j kuri laukia atminties sąrašo (tuple) žinutės iš praeitos gijos τi,j-1 turi žemesnį prioritetą ir taip toliau. pavyzdžiui gija kuri laukia žinutės iš prieš tai buvusios turi antrą aukščiausią prioritetą.

Gijos yra pertraukiamos ir numatomos iš anksto. Gijos bendrauja viena su kita per atminties sąrašus. Kai gija siunčia į atminties sąrašą, iššaukiamas bazinis kintamasis OUT, kad pakeistų gijos būseną iš laisvos arba sustabdytos į paruoštą. Taip pat yra perskaičiuojami laiko ruožai bei galutiniai terminai.

Taigi LIMOS yra resursus taupanti, gudri, realiu laiku dirbanti mikro operacinė sistema, kuri naudoja dviejų lygių įvykio ir gijos architektūrą. Dėl tos priežasties naudoja dviejų lygių tvarkaraščių sudarymo sistemą: viskas suplanuota iš anksto ir vienam įvykiui baigiantis pagal prioritetus parenkamas kitas. LIMOS apjungia TinyOS ir SDREAM privalumus. Ji veikia skirtingais rėžimais. Dviejų branduolių kombinacija gerokai praplečia aplikacijų galimybes, nuo vienos užduoties aplikacijų iki kelių užduočių [4].

1 lent. Programinės įrangos palyginimas

	Retos	**TinyOS**	**Limos**
Daugiagijė	yra		yra
Dinamiška (jutiklių skaičius gali kisti)	yra	yra	yra
Perkonfigūravimas (skirtingų modulių parinkimas)	yra	yra	yra
Atviro kodo		yra	
Asinchroninė (galimybė pradėti apdoroti duomenis nebaigus jų nuskaityti)		yra	

Mūsų sistemai labiausiai tinkama naudoti būtų TinyOS. Ji yra atviro kodo, kas leidžia mums ją laisvai naudoti. Taip pat ji turi didelę palaikančią bendruomenę, kuri nuolat ją tobulina ir kuria naujus modulius.

1.3 Duomenų perdavimo būdų analizė

Bevielių jutiklių tinkluose labai daug dėmesio turi būti skiriama duomenų perdavimui, nes tai yra viena daugiausiai energijos vartojančių šių tinklų dalių. Čia reikia atkreipti dėmesį ne tik į tai kokie protokolai, duomenų tipai bus vartojami, bet ir į paties tinklo struktūrą, jo poreikius tuos duomenis perduoti. Kuo mažiau duomenys bus siuntinėjami tuo mažiau energijos bus suvartojama, kuo mažiau mazgų turės tuos duomenis, tuo didesnė tikimybė juos prarasti. Taip pat labai svarbus yra perduodamų duomenų saugumas, čia reikia atsižvelgti į galimybę užkoduoti ir atkoduoti realiu laiku, taip pat tai, kiek papildomai duomenų prisideda prie siunčiamo paketo. Taip pat jei naudojamas kodavimas bus per silpnas, tada jis netenka prasmės.

Čedomir Stefanovič ir kiti (2010) savo darbe nagrinėja „bekoeficienčių“ („rateless“) paketų schemos energijos efektyvumą pasiskirsčiusiam duomenų laikymui bevieliuose jutiklių tinkluose. Darbe nemažai dėmesio kreipiama į saugius ir efektyvius duomenų perdavimo būdus. Vienį iš jų yra naudojant„fontano kodavimo“ („fountain code“) principus. Jie gana paprasti, patikimi ir efektyvūs.

Kodo koeficientas – tai yra naudingos informacijos ir papildomų simbolių, naudojamų jai atstatyti santykis. „Bekoeficienčiai“ kodai tai tokie kodai, kurie neturi fiksuoto santykio [8].

„Bekoeficienčių“ paketų pasiskirsčiusio laikymo veikimo principas remiasi atsitiktiniu bevielių jutiklių tinklų grafo perėjimu, todėl kartais gali pareikalauti labai daug ryšio ir energijos resursų. Visais kitais atžvilgiais jis panašus į centralizuoto laikymo principą. Šiame darbe ir buvo lyginami šie metodai pabrėžiant „bekoeficienčių“ paketų kodavimo privalumus.

Paprastai bevieliuose jutiklių tinkluose naudojamas centralizuotas metodas, kai visi gauti duomenys siunčiami vienam ar keliems iš anksto nustatytiems mazgams. Šie iš anksto nustatyti mazgai siunčia duomenis išoriniam tinklui, dažniausiai naudodami GPRS, kur šie duomenys yra toliau apdorojami ir laikomi. Šiame scenarijuje buvo nagrinėjamas atvejis, kai išorinis tinklas nėra prieinamas ir duomenys surenkami periodiškai iš pravažiuojančio automobilio ar praskrendančio lėktuvo. Taip pat nėra iš anksto nustatytų mazgų per kuriuos būtų surenkami duomenys, o patys mazgai yra pažeidžiami. Taip pat duomenys bus laikomi ne viename mazge, bet bus paskirstyti visame tinkle. Pageidaujama savybė duomenis surinkti aplankant tik kelis mazgus.

Nors duomenims laikyti siūloma daugybė metodų: duomenų išskaidymo, duomenų pasikartojimo skirtinguose mazguose, tačiau „bekoeficienčių“ paketų laikymo principas dėl savo efektyvumo energiją taupančiuose prietaisuose, pasirodė naudingiausias [5].

„Bekoeficienčiai“ kodai dar žinomi kaip „fontanų“ kodai, tai tokia kodų klasė, kurioje iš pradinės simbolių sekos gali būti sugeneruota begalė užkoduotų sekų. Sugeneruotos sekos turi savybę, kad praradus dalį duomenų, būtų galima atkurti pradinę seką jei turimos sekos simbolių kiekis yra lygus arba šiek tiek didesnis negu pradinių duomenų. Optimalus kodavimas gaunamas kai pradinę seką galima išgauti iš tokio paties simbolių skaičiaus kaip ir pradinė seka [6].

Pasiskirsčiusi duomenų saugojimo schema

„Bekoeficienčių“ paketų pasiskirsčiusio laikymo schemos tikslas yra tolygiai paskirstyti visus duomenis bevielių jutiklių tinkle. Pasiskirstęs „bekoeficientis“ kodavimas turėtų išlaikyti tas pačias savybes kaip ir centralizuotas (duomenų atstatymo galimybė). Visas kodavimo procesas padalintas į tris dalis: inicializacija, kodavimas ir paskirstymas.

Inicializacijos fazėje kiekvienas sensorius inicializuoja b skaičių „bekoeficienčių“ paketų, tai yra b savo duomenų paketo kopijų, Šie paketai yra išsiunčiami N skaičiui mazgų, taigi išviso gaunasi bN paketų tinkle. Kiekvienam iš b paketų yra priskiriamas laipsnis d atsitiktinai parinktas iš nustatytos pasiskirstymo aibės Ω(d). Šis laipsnis rodo kiek duomenų reikės surinkti iki kol visą turinį bus galima padėti į kurį nors mazgą. Šis laipsnis yra įrašomas į „bekoeficienčio“ paketo antraštę. Tų mazgų, kurių duomenys yra įrašomi į „bekoeficientį“ paketą, identifikacinis numeris taip pat yra įrašomas į antraštę, į atskirą lauką. Antraštėje taip pat yra atskiras laukas duomenų apjungimo laikui T_D, tai yra skaičius kiek paketas turi keliauti, prieš apjungdamas duomenis. Šis parametras reikalingas tam, kad būtų galima reguliuoti išsimaišymą tarp mazgų, kurių duomenys bus apjungiami.

Po inicializacijos fazės seka kodavimas. Šioje fazėje kiekvienas „bekoeficientis“ paketas turi prijungti d-1 skaičių duomenų paketų. Šie paketai parenkami „bekoeficienčiam“ paketui atsitiktinai keliaujant tinklu, sekantis mazgas parenkamas iš kaimyninių mazgų paketui atvykus. Paketui keliaujant tinklu, jis yra apdorojamas kiekviename mazge. Jei paketo duomenų apjungimo laikas yra didesnis nei nulis, tai mazge jis sumažinamas vienetu ir keliauja į kitą atsitiktinį mazgą. Jei duomenų apjungimo laikas baigėsi, mazgas įrašo savo duomenis į paketą, įrašo savo identifikacinį numerį į antraštę (lauką, kur surašomi mazgų, kurių duomenys buvo įrašyti, identifikaciniai numeriai), sumažina laipsnį d vienetu, atstato apjungimo laiką į pradinį ir pasiunčia paketą į kitą atsitiktinį mazgą. Išimtis yra daroma tada kai apjungimo laikas baigiasi tame mazge, kurio duomenys jau yra įrašyti į tą paketą, tada paketas tęsia savo kelionę tol kol nesutinka mazgo, kurio duomenų dar neturi. Kuo mažesnis maišymosi laikas, tuo daugiau papildomų šuolių reikia, kuo didesnis, tuo mažiau.

Paskutinė fazė yra išsklaidymas. Kai surenkami d - 1 skaičiaus mazgų duomenys, tada paketas pasiunčiamas į atsitiktinį mazgą. Tai daroma pasiunčiant paketą per papildomą T_D.

Idealiu atveju, jei paketas keliauja tik per nepasikartojančius mazgus, jo kelionės ilgis bus:

$$l` = d` \cdot T_D$$

d` - vidutinis paketų išsisklaidymo laipsnis.

Tada visų paketų kelias būtų:

$$L_D = bNl`$$

Tarkime, kad duomenų surinkimui bus naudojamas mobilus surinkėjas, kuris rinkimą pradės atsitiktiniame taške ir keliaus pro mazgus stengdamasis išvengti jau aplankytų vietų. Jis tęs savo kelionę tol, kol bus surinktas nustatytas duomenų kiekis. Kadangi naudojamas pasiskirstęs duomenų saugojimo metodas ir duomenys yra pasiskirstę plačiai po visą tinklą, nėra reikalo apeidinėti visus mazgus, nes tą pačią informaciją galima rasti keliose vietose.

Centralizuota duomenų saugojimo schema

Čia centralizuotas metodas naudojamas tik palyginimo tikslams ir padarytas taip, kad turėtų tą patį rezultatą kaip ir pasiskirstęs metodas. Kitaip tariant užkoduoti paketai turėtų būti pasiskirstę taip pat kaip ir pirmuoju atveju ir turėti lygiai tokį patį šansą išsaugoti duomenis. Taip pat duomenys bus surenkami taip pat, kaip ir pirmuoju atveju. Kiti reikalavimai, keliami centralizuotam metodui yra: paprastumas, duomenų persiuntimas, turint duomenis, tik apie kaimynus, taip pat reagavimas į tinklo dydžio pokyčius.

Šiuo atveju visą darbą atliks vienas mazgas, jis surinks visus duomenis, juos užkoduos ir persiųs saugoti kitiems mazgams. Šis kodavimo mazgas yra pagrindinis tinkle, kadangi visa informacija yra tvarkoma jo pagalba. Taip pat šis mazgas nėra žinomas iš anksto ir visi keliai turi būti surinkti. Tai padaroma paleidžiant daugybę paketų tinkle ir pagal jų kelius sudarant tinklo žemėlapį, mazgų skaičius šiuo atveju naudojamas kaip kelio kaina [7].

Po to kai koduojantis mazgas iš kitų mazgų gauna duomenis, jis juos užkoduoja ir persiunčia taip pat kaip ir pirmuoju atveju: naudojant atsitiktinius kelius. Kadangi šiuo atveju visi paketai keliauja iš vieno mazgo, naudoti vieno jungimosi laiką T_D nepakanka, norint, kad paketai pasiskirstytų tolygiai.

Šiuo atveju paketų kelionės ilgis yra:

$$L_C = N + h`N + bN \cdot T_C$$

h` - vidutinis kelias nuo mazgo iki koduojančio mazgo.

Palyginimo rezultatai

Gauti simuliacijos rezultatai parodė, kad didinant duomenų apjungimo laiką rezultatai gerėja tiek centralizuotos, tiek pasiskirsčiusios schemos atveju. Centralizuotas metodas yra kur kas efektyvesnis sunaudojamos energijos atveju, kadangi pasiskirsčiusios schemos atveju reikia labai daug paketų perdavimų paketų apjungimui. Tačiau centralizuotos schemos trūkumai neatperka jos privalumų, pasiskirsčiusi schema yra kur kas saugesnė ir atsparesnė mazgų gedimams. Centralizuotos schemos atveju nors vienam mazgui sugedus visą tinklo žemėlapį tektų sudarinėti iš naujo, taip reikėtų papildomų energijos atsargų, taip pat energijos suvartojimas tokiame tinkle nebūtų tolygus ir koduojantis mazgas, bei mazgai esantys arčiau jo greičiau išnaudotų savo energijos atsargas [5]. Taigi priklausomai nuo to kaip bus organizuojamas tinklo darbas, labai priklauso sistemos gyvavimo laikas.

Mūsų darbo atveju reikėtų naudoti hibridinę sistemą. Jei mazgas pajunta pavojų: gaisras, potvynis ar įsibrovimas, signalas būtų siunčiamas į mazgus link pagrindinio priėmimo įrenginio. Gavus tokį signalą būtų imtasi atitinkamų veiksmų. Ši schema panašesnė centralizuotai. Jei norime kaupti duomenis, tarkim kurioje vietoje ir kiek kartų buvo įsibrauta į tam tikras zonas arba kiek kartų ir kurios vietos buvo užpilamos, čia būtų geriau naudoti pasiskirsčiusią schemą, nes taip paketų perdavimo metu, visų mazgų baterijos būtų apkraunamos vienodai.

1.4 Energijos suvartojimo bevieliuose jutiklių tinkluose analizė

Bevieliuose jutiklių tinkluose energijos taupymas yra vienas iš prioritetinių uždavinių, nes nuo to priklausys visos sistemos darbo laikas. Taupoma yra visur, tiek techninėje tiek programinėje šios sistemos dalyje, todėl norint pasiekti maksimalų rezultatą, geriausia būtų šias dalis suderinti.

Pagrindiniai mazgo energiją naudojantys komponentai yra: sensoriaus modulis, procesorius, ryšio modulis ir energijos šaltinis. Sensoriaus modulis priklauso nuo stebėjimo objekto ir tinklo paskirties, tačiau dažniausiai ši dalis dažniausiai laikoma nesvarbia, nes vartoja labai nedaug energijos. Tačiau reikia atkreipti dėmesį į tai, kad kai kurių jutiklių

jautrumas priklauso būtent nuo suvartojamos energijos [9]. Daugiausiai energiją vartojančios dalys yra procesorius ir ryšio modulis. Ryšio modulio suvartojama energija labai priklauso nuo to kaip dažnai mazgams reikia komunikuoti vienas su kitu, ir nors techninė dalis yra svarbi, čia labiau reikia atsižvelgti į komunikacijos algoritmus ir visą sistemos struktūrą, bei valdymą. Kamal Beydoun ir Violeta Felea (2008) savo darbe [10] siūlo mažinti perdavimų skaičių. Kadangi suvartojamos energijos kiekis daugiau priklauso ne nuo signalo nukeliaujamo atstumo, o nuo šuolių skaičiaus, siunčiant duomenis kažkuria kryptimi galima būtų pašalinti tarpinius taškus ir siųsti toliausiai pasiekiamam mazgui.

Amit Sinha ir Anantha Chandrakasan (2001) savo darbe [11] analizuoja dinaminį galios valdymą. Procesorius ne visada turi dirbti maksimaliu rėžimu, kartais jo galią galima sumažinti. Tokie pokyčiai leidžia sutaupyti nemažai energijos. Taip pat mazgai, kai nedirba turėtų miegoti, tačiau mazgo pabudimas užima laiko, todėl tai gali trukdyti sistemos darbui. Labai svarbu parinkti tinkamus algoritmus, planuojančius sistemos darbą, kad šito būtų galima išvengti. Taip pat šie sistemos aspektai analizuojami ir Vijay Raghunathan bei kitų [12].

Dažniausiai mažiau dėmesio kreipiama į paties energijos šaltinio efektyvumą. Galios šuoliai jutiklio ir procesoriaus moduliuose, bei dinaminis galios valdymas dažniausiai padidina srovės svyravimus, o tai mažina energijos šaltinio efektyvumą. Shaoqiang Liu ir kiti (2009) savo darbe [9] rašo į ką reikėtų atkreipti dėmesį projektuojant energijos šaltinio modulį, kaip reikėtų suderinti su kitais komponentais ir jų darbu.

Kadangi bevielių jutiklių sistemose energijos taupymas yra būtina savybė, taupoma turi būti visose jo dalyse, tiek programinėje įrangoje, aktyviai valdant energijos vartojimą, tiek parenkant algoritmus duomenims perduoti, tiek techninėje dalyje. Maksimaliam rezultatui pasiekti netik reikia realizuoti taupymą šiose dalyse, bet ir suderinti jas tarpusavyje.

2. TINKLO STRUKTŪROS IR MODELIO SUDARYMAS

2.1 Tinklo struktūra, bei veikimo algoritmas

Pagrindinė šio tinklo užduotis yra pranešti apie galimą pavojų (gaisro, potvynio, įsibrovimo) . Visais stebėjimo atvejais tinklas turi būti suskirstytas į zonas. Tai reikalinga pavojaus vietos nustatymui. Jei vanduo apsėmė tik pirmąją pakopą, tai reaguoti nebūtina, jei labai greitai semia ir kitas, tada jau reikėtų imtis kažkokių priemonių: pranešti tarnyboms, išjungti prietaisus, jei yra galimybė nukreipti vandenį kita linkme. Visas reagavimas vykdomas pastate, todėl informacija iš vienos zonos turi būti perduodama per kitą, aukštesnio lygio zoną.

5 pav. stebėjimo zonų paskirstymas vandens jutiklių atveju

6 pav. Duomenų keliavimas iš vienos zonos į kitą, vandens jutiklių atveju

Taip pat kas kažkiek laiko turėtų būti pasiunčiamas patikrinimo signalas ar visi jutikliai veikia. Pagal šią schemą didžiausias krūvis tenka aukščiausio lygio zonai, todėl jos mazgus tektų atnaujinti dažniausiai.

Panašiai būtų ir gaisrų atveju, tačiau čia nėra zonų hierarchija egzistuoja tik informacijos perdavimui, nes gaisras gali kilti bet kurioje iš jų. Reaguojama atitinkamai nuo zonos. Priešingai negu potvynio atveju, jei nors vienas mazgas užfiksavo gaisrą, nesvarbu kurioje zonoje, būtina pranešti tarnyboms, nes ugnis greitai plinta ir gali persimesti į pastatus. Vienintelė vieta kurią reikėtų išskirti, tai pats pastatas, jei mazgas praneša apie gaisrą jame, reikia įjungti automatinį gesinimą.

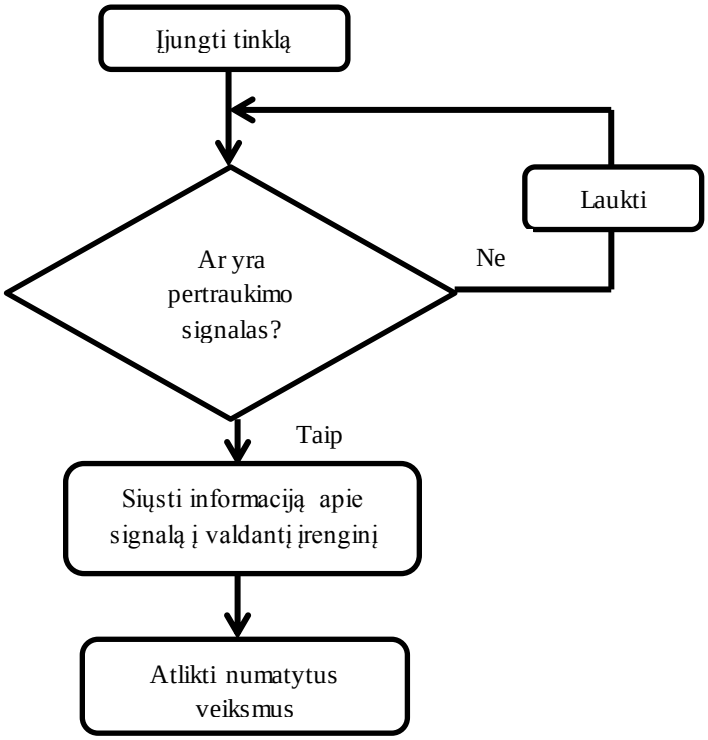
7 pav. Zonų paskirstymas gaisro jutiklių atveju

Apsaugos nuo įsibrovimo atveju, kaip ir potvynio, reikia skirstyti į zonas. Nors galima padaryti ir daugia, tačiau jas reikėtų skirstyti į tris pagrindines: lauko, kiemo ir vidaus. Apie judesius pastebėtus lauke reikėtų kaupti duomenis, tuo atveju jei kažkas „tyrinėja“ pastatą. Jei judesys užfiksuotas viduje, turi būti perduodamas signalas atitinkamoms tarnyboms. Kadangi mūsų tinklas yra netoli pastato, kuriame yra energijos šaltinis, bei visi prietaisai kurie reaguos į pranešimus, duomenų kaupimui apie įsibrovimus bus naudojama centralizuota schema.

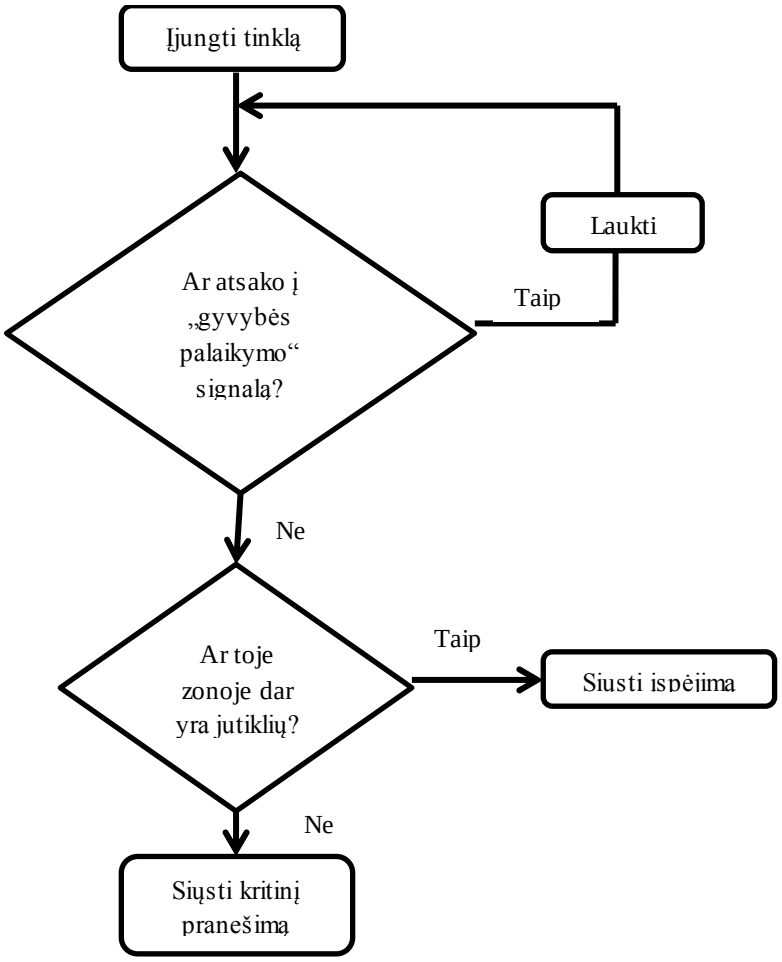
8 pav. Zonų paskirstymas judesio jutiklių atveju

Tinklo veikimo blokinės diagramos

9 pav. Bendra tinklo veikimo blokinė diagrama



10 pav. Tinklo mazgų tikrinimo blokinė diagrama



Šios blokinės diagramos tinka visiems atvejams. Skiriasi tik tai, kaip bus reaguojama.

2.2 Informacijos apdorojimo būdo parinkimas

Mazgui užfiksavus įvykį, informacija apie jį siunčiama iš to mazgo į centrinį kompiuterį. Tai daroma siunčiant iš zonos kurioje buvo užfiksuotas įvykis į aukštesnio lygiu zoną, kol nepasiekiamas galutinis įrenginys. Paketas susideda iš pradinio mazgo identifikacinio numerio, bei laiko, kada buvo pastebėtas įvykis. Šie duomenys išsaugomi centriniame mazge, bei priklausomai nuo įvykio, į jį reaguojama: susisiekiama su atitinkamomis tarnybomis, pranešama savininkui, išjungiami ar įjungiami atitinkami prietaisai.

11 pav. Informacijos srautas iki ją apdorojančio įrenginio

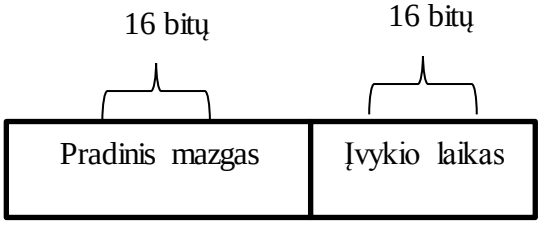
2.3 Informacijos kaupimo ir perdavimo būdo parinkimas

Kadangi informacija kaupiama, tik apie įvykius ir apie juos vis tiek pranešama centriniam įrenginiui, tai ir kaupiama ji bus jame. Be to centrinis įrenginys, būdamas pastate, yra saugesnis nei bet kuris mazgas. Taip pat pastate yra energijos šaltinis, todėl valdantis įrenginys gali netaupyti energijos. Gavus pranešimą apie įvykį jis tiesiog įrašomas į duomenų bazę.

Kadangi visa sistema naudoja zonas įvykiams fiksuoti ir vieną centralizuotą įrenginį informacijai kaupti, tai ir perdavimas bus nuo įvykio vietos link centro, perduodant iš vienos zonos į kitą.

Paketas užtektų tik kelių laukų: mazgo kuriame pastebėtas įvykis kodo ir laiko kada buvo pastebėtas. Kur paketą siųsti toliau spręstų pats mazgas, nes paketo galutinis tikslas visada yra valdantis įrenginys.

12 pav. Siunčiamo paketo struktūra



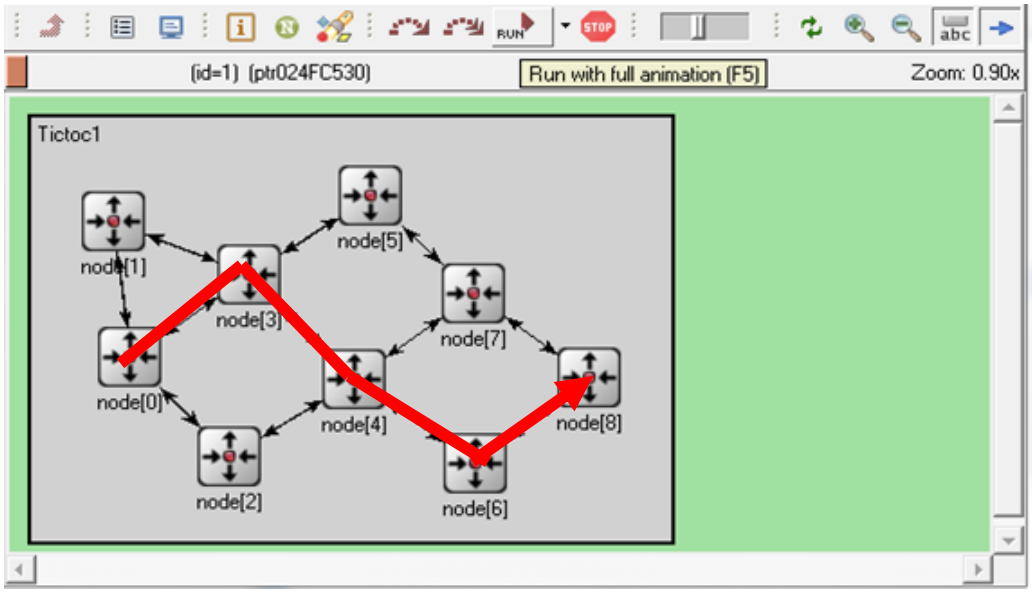
Įvykio laikui koduoti reikia 16 bitų paketo dalies, taip yra todėl, kad siunčiamas keturių skaitmenų dešimtainis skaičius. Pavyzdžiui 2346, tai reiškia, kad įvykis pastebėtas 23 valandą 46 minutės.

Mazgo identifikaciniam numeriui taip pat naudojamas 4 skaitmenų skaičius. Nors tinkle mazgų nėra daug, tačiau ilgalaikėje perspektyvoje, kai vieni mazgai keičiami kitais, dėl energijos šaltinio išsekimo, reikia suteikti naujus identifikacinius numerius. Taigi tinklo gyvavimo metu būtų galima turėti 9999 unikalius identifikacinius numerius.

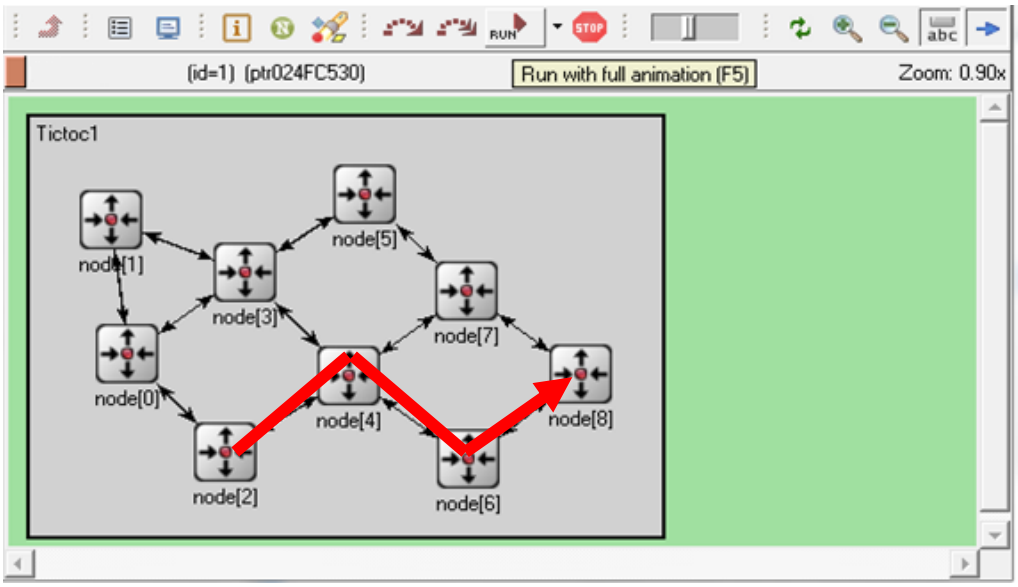
3. EKSPERIMENTAS

Tinkas buvo modeliuojamas naudojant OMNET++ programinę įrangą. Tai yra atviros architektūros tinklo modeliavimo programa, dažniausiai naudojama akademinės bendruomenės.

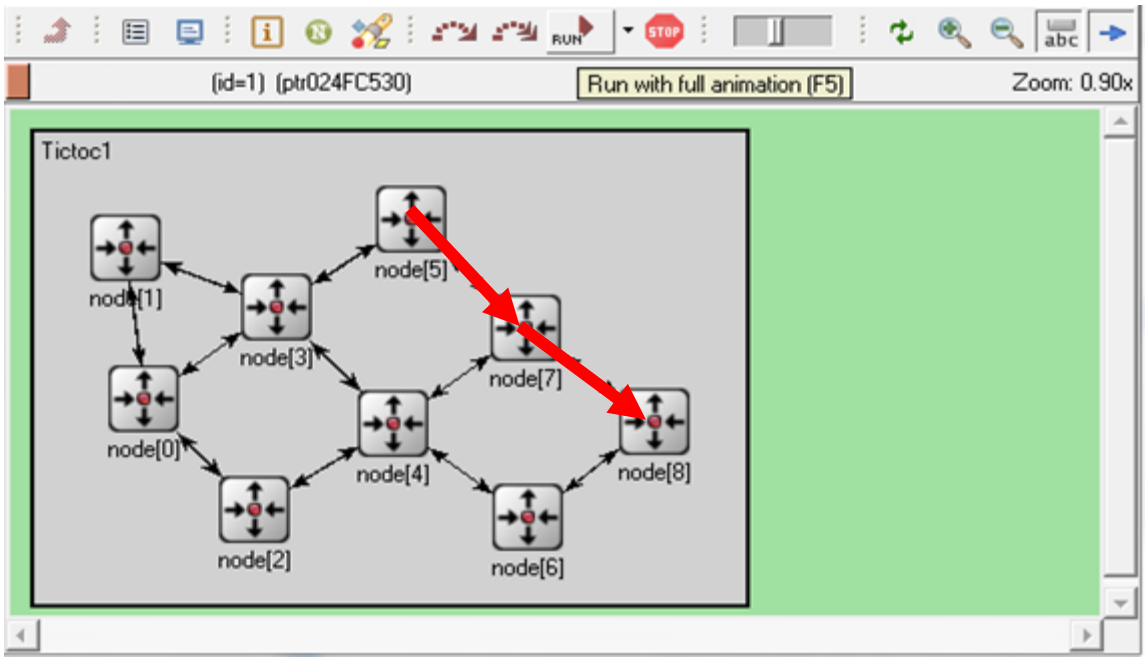
13 pav. Srauto schema tinkle, jei įvykis užfiksuojamas nuliniame mazge.



14 pav. Srauto schema tinkle, jei įvykis užfiksuojamas antrame mazge

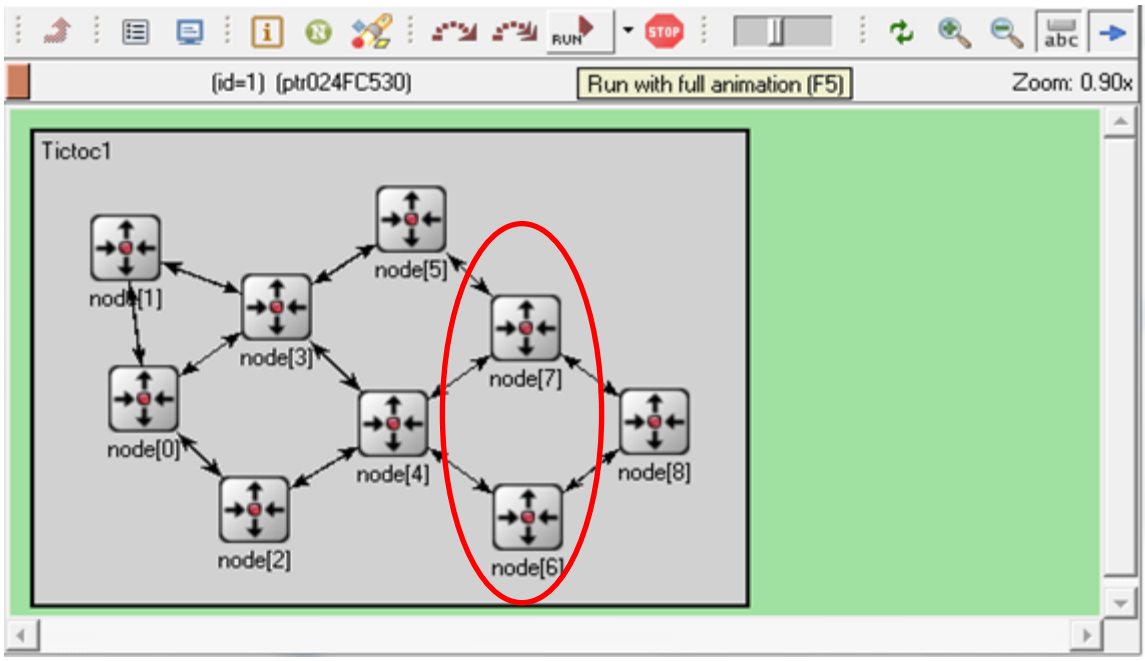


15 pav. Srauto schema tinkle, jei įvykis užfiksuojamas penktame mazge



Paketas keliauja iš mazgo, kuriame buvo pastebėtas įvykis link pagrindinio valdymo įrenginio.

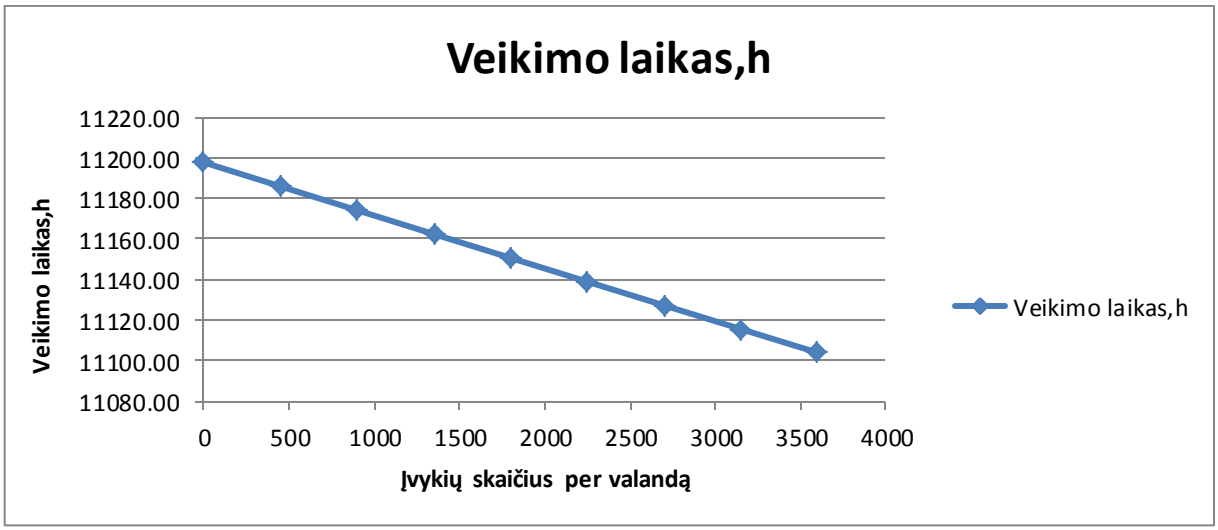
16 pav. Labiausiai apkraunami mazgai



Visais atvejais, nepriklausomai nuo to, kur pastebėtas įvykis, didžiausias apkrovimas, kaip ir buvo numatyta yra mazguose esančiuose ryšio atstume su valdančiu įrenginiu, šiame pavyzdyje tai yra šeštas ir septintas mazgai. Pats valdantis įrenginys (aštuntasis mazgas), turi pastovų energijos šaltinį, todėl jo atskirai nagrinėti nereikia.

Tarkime, kad naudosime Fujimax 2800 mAh AA bateriją. Jutiklį vandeniui aptikti MAXQ3210. Bei TELRAN TZ1053 itin mažai energijos suvartojantį radijo modulį.

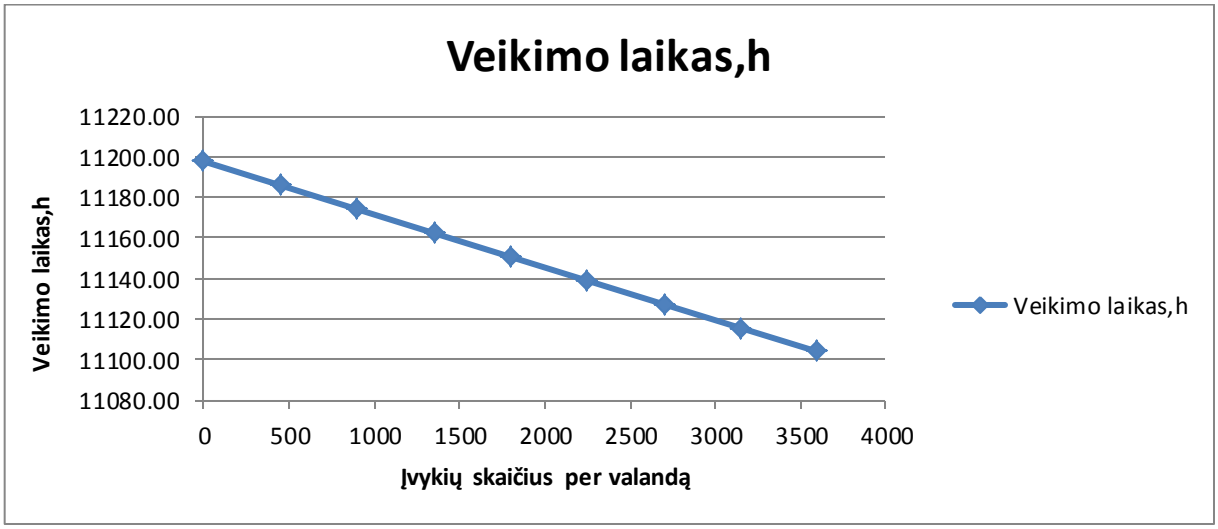
16 pav. Sistemos veikimo laikas



2 lent. Veikimo laiko skaičiavimo parametrai

Vartojama energija, mA	Atsibudimų skaičius per valandą	Atsibudimo trukmė, ms	Miego rėžimo trukmė, ms	Energijos vartojimas miego rėžime, mA	Energijos vartojimas siuntimui, mA	Išsiuntimo laikas, ms	Įvykių skaičius per valandą	Veikimo laikas,h
5	3600	50	950	0.000055	3.3	0.64	0	11086.26
5	3600	50	950	0.000055	3.3	0.64	450	11074.68
5	3600	50	950	0.000055	3.3	0.64	900	11063.13
5	3600	50	950	0.000055	3.3	0.64	1350	11051.60
5	3600	50	950	0.000055	3.3	0.64	1800	11040.10
5	3600	50	950	0.000055	3.3	0.64	2250	11028.62
5	3600	50	950	0.000055	3.3	0.64	2700	11017.16
5	3600	50	950	0.000055	3.3	0.64	3150	11005.73
5	3600	50	950	0.000055	3.3	0.64	3600	10994.32

17 pav. Sistemos veikimo laikas, be miego rėžimo



3lent.Veikimo laiko skaičiavimo parametrai, be miego rėžimo

Baterijos talpa, mAh	Vartojama energija, mA	Atsibudimų skaičius per valandą	Atsibudimo trukmė, ms	Energijos vartojimas siuntimui, mA	Išsiuntimo laikas, ms	Įvykių skaičius per valandą	Veikimo laikas,h
2800	5	3600	100	3.3	0.64	0	5600.00
2800	5	3600	100	3.3	0.64	450	5597.04
2800	5	3600	100	3.3	0.64	900	5594.09
2800	5	3600	100	3.3	0.64	1350	5591.14
2800	5	3600	100	3.3	0.64	1800	5588.20
2800	5	3600	100	3.3	0.64	2250	5585.25
2800	5	3600	100	3.3	0.64	2700	5582.32
2800	5	3600	100	3.3	0.64	3150	5579.38
2800	5	3600	100	3.3	0.64	3600	5576.45

Įvykių skaičius sistemai didelės įtakos nedaro, tačiau panaikinus miego rėžimą, sistemos gyvavimo laikas sutrumpėja beveik dvigubai .

4. IŠVADOS

- Buvo atlikta jutiklių analizė ir išrinkti būtiniausi jų parametrai, vandens, gaisro, bei judesio sistemoms. Vandens sistemai parinkta: atsparumas drėgmei, korozijai, bei galimybė įtvirtinti. Gaisro sistemai: atsparumas ugniai. Judesio sistemai privalumas yra judančio objekto dydžio nustatymas.
- Atlikta programinės įrangos analizė, buvo parinkta TinyOS operacinė sistema, dėl galimybės keisti modulius, galimybės dinamiškai keisti jutiklių skaičių veikiančioje sistemoje, atviro kodo, bei galimybės apdoroti duomenis nebaigus jų visų skaityti.
- Išanalizuoti duomenų kaupimo bei perdavimo būdai: pasiskirstęs duomenų kaupimo, bei perdavimo būdas yra pranašesnis sistemų, kuriose nėra galimybės įrengti vieną gerai apsaugotą mazgą atveju. Namo ir jo teritorijos stebėjimo atveju, dėl energijos suvartojimo ir būtinumo vienam įrenginiui apdoroti informaciją ir priimti sprendimus, tinkamesnis yra centralizuotas metodas.
- Atlikus energijos valdymo analizė, dėl nedidelių energijos atsargų parinktas taupymas visose mazgo dalyse. Nors vienai daliai vartojant daugiau energijos ženkliai krenta tinklo gyvavimo laikas.
- Buvo sudaryta tinklo struktūra bei modelis, nustatyta, kad miego rėžimas gali ženkliai prailginti sistemos veikimo laiką (žiūrėti 3 lentelę). Taip pat nustatyta, kad tokios sistemos gyvavimo laikas yra apie 458 dienas priklausomai nuo įvykių skaičiaus (žiūrėti 2 lentelę).

LITERATŪRA

1. Informacija apie bevielių jutiklių tinklus [žiūrėta 2010-10-12]. Prieiga per internetą < http://en.wikipedia.org/wiki/Wireless_sensor_network/>.
2. CULLER, D. TinyOS: Operating System Design for Wireless Sensor Networks. Journal of Infrastructure Engineering, 2008, p. 89-101.
3. HOJUNG, Cha, et al. RETOS: Resilient, Expandable, and Threaded Operating System for Wireless Sensor Networks.Tarptautinės konferencijos pranešimų medžiaga. Niujorkas, 2007.
4. HAI-YING Z. LIMOS: a Lightweight Multi-threading Operating System dedicated to Wireless Sensor Networks. Tarptautinės konferencijos pranešimų medžiaga. Šanchajus, 2007.
5. ČEDOMIR, Stefanovič, et al. On Energy Efficiency of Rateless Packet Scheme for Distributed Data Storage in Wireless Sensor Networks. Tarptautinės konferencijos pranešimų medžiaga. Kranjaska Gora, 2010 .
6. Informacija apie fontanų kodus [žiūrėta 2010-11-12]. Prieiga per internetą <http://en.wikipedia.org/wiki/Fountain_code/>
7. FAN, Ye, et al. A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks. Tarptautinės konferencijos pranešimų medžiaga. Skotsdeilas, 2001.
8. Informacija apie naudingos informacijos kiekį kode [žiūrėta 2010-11-15] <http://en.wikipedia.org/wiki/Code_rate>
9. SHAOQIANG, Liu, et al. Scheme for High Power Supply Efficiency of WSN Node. Tarptautinė konferencijų medžiaga. Beidžingas, 2009.
10. BEYDOUN, K.; ir FELEA, V. Energy-Efficient WSN Infrastructure. Tarptautinės konferencijos medžiaga. Irvinas, 2008, p. 58-65.
11. SINHA, A.; ir CHANDRAKASAN, A. Dynamic Power Management in Wireless Sensor Networks. Design & Test of Computers, 2002, nr. 18, p. 62-64.
12. RAGHUNATHAN, Vijay, et al. Energy-Aware Wireless Microsensor Networks. IEEE signal processing, 2002, p. 40-50.

RESEARCH AND DEVELOPMENT OF REAL TIME ENVIRONMENT MONITORING SYSTEM

SUMMARY

Information is one of the most valuable assets. Today it is being gathered almost everywhere. Only with the correct information it is possible to make the right decisions.

The problem is that there are none cheap, effective and fast deployable way to monitor the environment. There are several popular ways to gather it. One way is to do it using human resources. This way is simple, but if the monitoring area is large or if it needs to be monitored often, the cost of it becomes larger than automated systems. Other way is for monitoring environment is adding stations with internet connections, but these systems are hard to deploy since there is a of internet cables for every station.

The purpose of this work is to do a research that consists of wireless sensors analysis, node software analysis, data transmission analysis and energy consumption evaluation. Develop a system model that could solve this problem.

Terminų ir santrumpų žodynas

WSN – bevielių jutiklių tinklas

MAC – unikalus tinklo įrenginio adresas