Opis i primena softverskih alata u programiranju bežičnih senzorskih mreža

Branislav Novkov, Miladin Jovanović

***Sadržaj* — U ovom radu su prezentovani osnovni koncepti bežičnih senzorskih mreža i upotreba softverskih alata kroz praktične primere. Prikazani su principi operativnog sistema TinyOS i programskog jezika nesC kao namenskih softverskih alata koji se koriste za programiranje aplikacija na bežičnim senzorima, kao i java aplikacije kroz koje je moguće pratiti ponašanje bežične senzorske mreže.**

***Ključne reči* — Bežične senzorske mreže, nesC, TinyOS.**

I. UVOD

IAKO se bežične ad-hok mreže proučavaju već tri decenije, bežične senzorske mreže predstavljaju novu tehnologiju koja se intenzivno razvija poslednjih godina. Bežične senzorske mreže su namenske mreže koje se sastoje od mnoštva senzora koji mogu da pokriju velika područja i obezbede praćenje i dostavu željenih informacija. Ove mreže su prvo razvijane i korišćene za vojne upotrebe, a u poslednje vreme se koriste i za civilne svrhe. Generalno, osnovna upotreba bežičnih sensorskih mreža je za nadzor, odnosno praćenje određenih fenomena, kao što su vremenski uslovi, nadzor okoline, medicinski nadzor, detekcija seizmičkih aktivnosti, vojno praćenje i mnogi drugi.

Obično kod bežičnih senzorskih mreže ne postoji prava hijerarhijska struktura i svaki čvor, koji se još naziva i mot (*mote*), komunicira sa bilo kojim susedom koji se nalazi u njegovom radio opsegu. Čvorovi mreže moraju biti u stanju da identifikuju svoju okolinu i na osnovu dinamičkih algoritama da osvežavaju tabele rutiranja, uz čiju pomoć će odrediti najbolji put za slanje informacija. Takođe, mreža treba da poseduje osobinu samokonfigurisanja (*self-confguring*), tj. čvorovi moraju biti u stanju da se samostalno organizuju u mrežu i da samostalno održavaju komunikacione puteve. Takođe, u slučaju da određeni čvor prestane sa radom, njegovi susedi moraju biti u stanju da za vrlo kratko vreme nađu alternativnu putanju, tj mreža treba da poseduje osobinu samooporavljanja (*self-healing*) [1], [2].

S druge strane, veličina senzorskog čvora mora biti mala u cilju što lakšeg raspoređivanja po željenoj oblasti. Čvorovi moraju imati što manji utrošak energije, tj. eksploatacije mreže treba da bude što duža bez potrebe za zamenom baterija. Takođe, izrada, programiranje i održavanje čvorova mreže treba da budu što je moguće jeftinije jer ove mreže mogu sadržati na hiljade čvorova. Iako resursno ograničeni, čvorovi u mreži treba da budu u stanju da izvršavaju relativno složene procese. Usled svih ovih zahteva bežične senzorske mreže su zaokupirale pažnju tehnološke javnosti.

Iako postoji mnogo aplikacija u kojima se bežične senzorske mreže već koriste, još nisu u potupnosti standardizovani komunikacioni protokoli za ovaj tip mreža. Kao standard se nameće ZigBee protokol stek, čiji su donji nivoi definisani IEEE 802.15.4 protokolom [3], [4]. Naime, IEEE 802.15.4 definiše nivo voda podataka i fizički sloj OSI modela za nisko-protočne bežične personalne mreže (*LRWPAN–Low-Rate Wireless Personal Area Network*). ZigBee definiše mrežni i aplikacioni sloj OSI modela.

Naredni tekst organizovan je u sledeće celine. U drugom odeljku dat je kratak opis opreme koja se koristi za realizaciju praktičnih primera. Treći i četvrti odeljak su posvećeni principima operativnog sistema TinyOS i programskog jezika nesC. U petom odeljku su opisani praktični primeri, a u poslednjem odeljku dat je zaključak.

II. KARAKTERISTIKE HARDVERSKIH KOMPONENTI

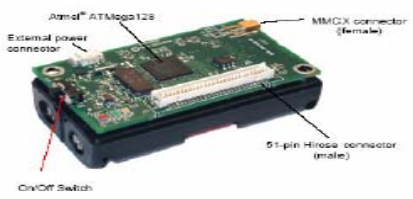
Bežične senzorske mreže su sastavljene od velikog broja malih, resursima ograničenih uređaja–motova. Mot je sastavljen iz četiri dela. Prvi je niskoenergetski mikoroprocesor i memorija koji obavljaju procesiranje podataka i logičke zadatke. Drugi deo predstavlja senzorska komponenta koja registruje fizičke promene u okolini i konvertuje analogne signale koje je registrovala u digitalne putem A/D konvertora. Treći deo predstavlja primopredajnik, koji. može biti realizovan kao optički (laser), infracrveni ili radio primopredajnik. Četvrti deo predstavlja jedinica za napajanje. Ona se najčešće realizuje pomoću baterija, ali postoje i motovi koji se mogu sami napajati iz okoline (putem solarne ili vibracione energije). Mikroprocesor, koji predstavlja jezgro mota, nadgleda jedan ili više senzora koji su na njega prikačeni i od kojih dobija podatke o okolini, i pomoću primopredajnika komunicira sa spoljašnjim svetom.

U praktičnim primerima prikazanim u ovom radu se koristi Crossbow razvojno okruženje [5], [6], koje se sastoji iz sledećih komponenata:

B. Novkov, FTN, Trg D. Obradovića 6, 21000 Novi Sad, Srbija, (telefon: 381-64-1840344; e-mail: branislavnovkov@gmail.com).

M. Jovanović, FTN, Trg D. Obradovića 6, 21000 Novi Sad, Srbija, (telefon: 381-63-7625810; e-mail: malinadin@hotmail.com).

A. MPR410 (MICA2) mot (Sl 1)



Sl. 1. MICA2 mot

Osnovne karakteristike ovog mota su:

- Procesorski čip ATMega 128L. Čip je 8-mo bitni, radne frekvencija od 7.37 MHz.
- Programska memorija veličine 128 Kb (4 Kb SRAM-a).
- 51-pinski muški konektor sa 10-bitnim A/D konvertorom koji služi da se ploča priključi na programator ili da se na njega priključi senzorska ploča.
- Radio antena CC1000 sa FSK modulacijom i radnom frekvencijom od 433 MHz.

B. MPR510 (MICA2DOT) mot (Sl.2.)

Sl. 2. MICA2DOT mot

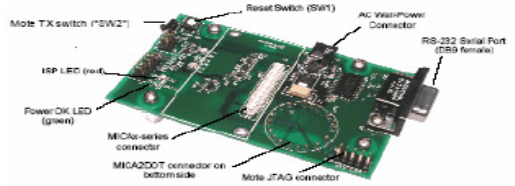
Osnovne karakteristike ovog mota su:

- Procesorski čip ATMega 128L. Čip je 8-mo bitni, radne frekvencije od 4 MHz.
- Programska memorija veličine 128 Kb (4 Kb SRAM-a).
- 18-pinski muški konektor sa 10-bitnim A/D konvertorom koji služi da se ploča priključi na programator ili da se na njega priključi senzorska ploča.
- Radio antena CC1000 sa FSK modulacijom i radnom frekvencijom od 433 MHz.

C. MIB510 serijska interfejs ploča (serial interface board) (Sl.3.)

Osnovne karakteristike ove ploče, koja služi za programiranje motova su:

- Procesorski čip ATMega 16L.
- Konektor za eksterno napajanje putem adaptera (5-7 V).
- RS-232 serijski port (ženski) koji služi za komunikaciju sa računarom.
- 51-pinski ženski konektor za priključivanje MICA2 mota i 19-pinski konektor za priključivanje MICA2DOT mota.

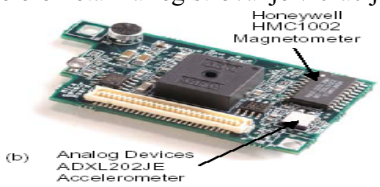


Sl. 3. MIB510 serijska interfejs ploča

D. MTS300 senzorska ploča (Sl.4.)

Njene osnovne karakteristike su:

- 51-pinski ženski konektor koji služi da se ploča prikači na MICA2 mot.
- Mikrofon sa piezoelektričnim rezonatorom i radnom frekvencijom od 4KHz.
- Senzori za svetlost (CdSe fotoćelije) i temperaturu (radni opseg od -40 do 70 ºC).
- Magnetometar za registrovanje magnetnog polja i 10-bitni akcelerometar za registrovanje vibracija.



Sl. 4. MTS310 senzorska ploča

III. OPERATIVNI SISTEM TINYOS

TinyOS [2], [7] je operativni sistem specijalno dizajniran za bežične sensorske mreže. Za razliku od tradicionalnih računara, motovi se koriste za prikupljanje podataka i kontrolu okoline u kojoj se nalaze. Iako ograničenih resursa, moraju biti sposobni da reaguju na sve događaje i da vrše obradu prikupljenih podataka. Iz ovoga proizilaze dva suštinska koncepta koji će odraziti u dizaniranju operativnog sistema. Prvi je da motovi reaguju na događaje iz spoljne okoline i pokreću se tim istim događajima (*event-driven*). Događaji su npr. dolazak poruke ili podataka sa senzora. Drugi koncept proizilazi iz činjenice da su reagovanja i manipulisanja događajima od strane mota s jedne strane, i procesiranje podataka s druge strane, dve konkurentne aktivnosti i samim tim mora postojati način kontrole ove konkurencije u cilju sprečavanja potencijalnih grešaka.

Programski model TinyOS-a je napravljen upravo da omogući realizaciju aplikacija koje se pokreću događajima. Kao namenski operativni sistem on reaguje na hardverske događaje, rukuje njima i podržava koncept taskova (*task*) koji su ekvivalentni funkcijama u drugim programskim okruženjima. Slede dve osnovne osobine TinyOS operativnog sistema koje formiraju programsku filozofiju koja će se realizovati kroz programski jezik nesC:

A. Arhitektura zasnovana na komponentama

Svaka aplikacija zasnovana na TinyOS operativnom sistemu sastavljena je od komponenti sa svojim interfejsima. Komponente predstavljaju strukturne elemente svake aplikacije. Komponenta može da pruža usluge preko svog interfejsa ali i da traži usluge preko nekog drugog interfejsa. Komponente se realizuju putem modula i povezuju putem konfiguracija. U modulima se definiše programski kod koji određuje funkcionalnost komponente i implementacija interfejsa te komponente. Interfejsi su jedine pristupne tačke komponenti i oni definišu njihovo ponašanje. U interfejsima je naveden skup funkcija koje se zovu komande, koje moraju biti implementirane u komponenti koja pruža interfejs i skup funkcija po imenu događaji koje moraju biti implementirane u komponenti koja koristi interfejs. Konfiguracija je komponenta koja povezuje druge komponente spajajući na taj način interfejse korišćene od strane jedne komponente sa interfejsima koje pružaju

druge komponente, što se naziva povezivanje (*wiring*).

Osnovna ideja je da se aplikacija napravi od modula i da se oni povežu putem konfiguracije. Pošto je povezivanje nezavisno od realizacije komponente na ovaj način je otvoren put inovacijama i implementacijama uz smanjenje veličine koda aplikacije, što se i zahteva usled ograničene količine memorije u senzorskim mrežama. Svaka aplikacija poseduje glavnu konfiguraciju koja povezuje glavni modul aplikacije sa ostalim komponentama. TinyOS sistemska biblioteka sadrži, između ostalog, mrežne protokole, servise i alate za skupljanje podataka sa senzora. Interfejsi ovih komponenti su otvoreni na korišćenje svim drugim komponentama.

B. *Model konkurencije*

Postoje dva načina izvršavanja koda aplikacije i oni predstavljaju izvore konkurencije u okviru TinyOS-a. To su taskovi i događaji. Taskovi su funkcije čije se izvršavanje odlaže praveći se tako raspored taskova. Kad jednom dođu na red oni se izvršavaju do kraja i ne prekidaju izvršavanje drugih taskova. Događaji se izvršavaju kao odgovor na hardverski prekid i takođe se izvršavaju do kraja, ali događaji mogu da prekinu izvršavanje taska ili nekog drugog događaja. Ovim se mogu stvoriti uslovi za trku (tzv. *race-conditions*) pri izvršavanju programa, što može dovesti do grešaka, kao što je npr. nerealizovanje zahtevane komande na vreme.

Interakcija sa operativnim sistemom i opremom koja se koristi u vežbama vrši se u okviru Cygwin okruženja koje dolazi uz instalaciju TinyOS operativnog sistema.

IV. PROGRAMSKI JEZIK NESC

Kompletan TinyOS sistem, njegove biblioteke i aplikacije su implementirane u programskom jeziku nesC. NesC jezik je namenski programski jezik koji ima sintaksu sličnu sintaksi programskog jezika C. Dizajniran je u cilju zadovoljavanja koncepta koji proizilazi iz strukture TinyOS operativnog sistema. Njegove osobine su:

- Kroz nesC je implementiran model konkurencije i izvršavanje aplikacija putem događaja. Realizovan je mehanizam za pravljenje, imenovanje i povezivanje komponenata potrebnih za realizaciju aplikacije zadovoljavajući na taj način programersku filozofiju TinyOS operativnog sistema.
- Predstavljajući proširenje programskog jezika C, uz pomoć nesC-a je moguće napisati efikasne aplikacije za mikrokontrolere na motovima i takođe su uzeti svi neophodni alati za interakciju sa hardverom. NesC je eliminisao nedostatak strukturiranja koda putem koncepta komponenti i detaljnom analizom pri kompajliranju.
- Usled ograničene veličine koda, nesC kompajler vrši njegovu analizu u celosti, nema kompajliranja koda u delovima. Kompajler prijavljuje sve moguće uslove trke u izvršenju programa povećavajući na taj način sigurnost. Pri kompajliranju vrši se detaljna analiza programa u cilju pouzdanosti i optimizacije programa. Smanjuje se veličina koda i vrši se eliminacija nepotrebnih delova koda.
- NesC je statički jezik. Nema dinamičke alokacije memorije i kompletan dijagram poziva je poznat pri kompajliranju. Kompajler pravi statičke instance potrebnih interfejsa. Ove restrikcije omogućavaju analizu programa i čine optimizaciju znatno jednostavnijom i preciznijom.

V. PRAKTIČNI PRIMERI

U ovom odeljku su opisani praktični primeri koji pokazuju principe istaknute u prethodnim odeljcima.

U prvoj primeru ilustruje se aplikacija pod nazivom Testled, koja omogućava da crvena led dioda na MICA2 motu treperi svake sekunde. Aplikacija se sastoji iz dve komponente. Prva je modul TestledM.nc u kome je realizovana aplikacija. Druga komponenta je konfiguracija Testled.nc koja i predstavlja glavnu konfiguraciju aplikacije i služi da se modul TestledM.nc poveže sa drugim sistemskim komponentama koje ova aplikacija zahteva za svoje izvršavanje. Kod konfiguracije TestledM.nc je sledeći:

```
configuration Testled{
}
implementation{
  components Main, TestledM, SingleTimer, LedsC;
  Main.StdControl->TestledM.StdControl;
  Main.StdControl->SingleTimer.StdControl;
  TestledM.Timer->SingleTimer.Timer;
  TestledM.Leds->LedsC;
}
```

Konfiguracija referencira četiri komponente. To su Main, TestledM, SingleTimer i LedsC. SingleTimer služi za kontrolu brojača na MICA2 motu dok LedsC služi za kontrolu dioda. Sledi povezivanje interfejsa koje koriste jedne komponente sa interfejsima koje druge komponente pružaju. StdControl je standardni sistemski interfejs koji služi za inicijalizaciju, startovanje i zaustavljanje komponenti. Main komponenta je prva koja će se izvršiti pri pokretanju bilo koje TinyOS aplikacije, tako da će se prvo inicijalizovati i startovati potrebne komponente. Sintaksa NesC-a govori da će se StdControl interfejs Main komponente povezati sa StdControl interfejsom kako modula TestledM tako i komponente SingleTimer. Smer strelice uvek ide od komponente koja koristi interfejs prema komponenti koja ga pruža. Strelica može biti okrenuta i u drugu stranu ali mora biti zadovoljeno pravilo navedeno u prethodnoj rečenici.

Pogledajmo deo realizacije modula TestledM:

```
event result_t Timer.fired()
{
  call Leds.redToggle();
  return SUCCESS;
}
```

Ovde je realizovan događaj isteka brojača. Kao što je rečeno već u prethodnim odeljcima komponenta koja koristi interfejs mora implementirati događaj koji se nalazi u korišćenom interfejsu. Svaki put kada brojač otkuca zadato vreme poziva se komanda za paljenje crvene led diode. Na istom principu se mogu uzimati periodična očitavanja sa senzora. U ovom slučaju se pri

implementaciji događaja isteka brojača koriste odgovarajuće naredbe potrebne za operaciju očitavanja sa senzora. Očitavanja sa senzora se vrše pomoću sistemskih komponenti.

Sledeći primer ukratko ilustruje kako je moguće realizovati radio komunikaciju između dva mota. Jedan MICA2 mot je isprogramiran da nakon isteka brojača, vrednost brojača (tri najveća bita) prikaže na led diodama i da tu istu vrednost emituje dalje putem radio interfejsa. Drugi mot je isprogramiran da prima vrednosti sa radio interfejsa i prikazuje ih na svojim led diodama. Za realizaciju aplikacija za oba mota potrebne su samo komponente koje se nalaze u sistemskoj biblioteci. Zahteva se samo realizacija konfiguracije koja će njih sve povezati u funkcionalnu celinu i realizacija svih potrebnih događaja. Kao što je već napomenuto u prethodnim odeljcima, povezivanje komponenti je nezavisno od njihove realizacije, što otvara mogućnost da se različite aplikacije realizuju upotrebom istih komponenti. Potrebno je samo komponente povezati na pravi način.

Napomenućemo da TinyOS okruženje nudi mogućnost simuliranja i debagovanja aplikacija. Simulator ne radi sa opremom nego pravi izvršni dokument na računaru uz pomoć koda same aplikacije koju treba da se simulira. Izdavanjem odgovarajućih naredbi u cygwin okruženju omogućeno je detaljno praćenje izvršavanja programa, postavljanje prekidnih tačaka, praćenje željenih funkcija ili promenjivih... Pošto ne radi sa opremom simulator ne može da modeluje okolinu u kojoj je oprema postavljena (ne uračunava kvalitet veze, snagu signala na izlaznom radio interfejsu...) ali je idealan za simulaciju TinyOS okruženja i otklanjanja mogućih grešaka koje su nastale usled toga.

Uz instalaciju TinyOS okruženja, pored Cygwin-a dolaze i java alati potrebni za interakciju računara i motova putem serijske interfejs ploče. Poslednji primer ilustruje upotrebu ovih alata. Prvo se koristi aplikacija *Listen* koja uzima podatke koji dolaze sa serijskog porta i prikazuje ih u heksadecimalnom obliku u Cygwin okruženju. Programiranjem MICA2 ploče da uzima očitavanja sa senzora i šalje ih na serijski port, moguće je vizuelno pratiti i očitavati vrednosti sa senzora. Naravno, u sistemskoj biblioteci TinyOS okruženja postoje realizovane komponente za interakciju mota i serijskog porta. Sledeća aplikacija je *SerialForwarder* koja takođe komunicira sa serijskim portom ali primljene pakete ne prikazuje već ih kopira i šalje preko TCP porta svim drugim aplikacijama kojima su oni potrebni. Broj kopija zavisi od broja aplikacija koje ih zahtevaju. Ovom aplikacijom je omogućeno korišćenje serijskog porta od strane više programa odjednom. Realizovana je i aplikacija pod nazivom *Oscilloscope* koja služi za vizuelno praćenje očitavanja sa senzora putem 2-D grafika. Ova aplikacija preko SerialForwarder-a dobija pakete i na x-osi prezentuje broj primljenog paketa, dok je na y-osi vrednost očitavanja u decimalnom zapisu.

Dodajmo na kraju da se u sistemskoj biblioteci takođe nalaze realizacije protokola za pristup medijumu i rutiranje, konkretno S-MAC (*Sensor Medium Access Control Protocol*) [8] i Xmesh [9]. Uz pomoć njih je moguće realizovati kompletnu mrežu koja može da se koristi u nadzoru neke željene oblasti.

VI. ZAKLJUČAK

Razvoj bežičnih senzorskih mreža u zadnjih nekoliko godina pokazuje da će ovakve mreže imati veliku perspektivu i brojne primene. Sa druge strane, zahtevi i ograničenja koji se pred njih nameću čine specifičan inženjerski problem, za koje još uvek ne postoje standardizovana i opšte prihvaćena rešenja. Praktični primeri, odnosno koncepti i osobine bežičnih senzorskih mreža koje su prezentovane u ovom radu mogu poslužiti kao osnova za upoznavanje sa ovom vrstom problematike.

LITERATURA

[1] V. Salazas, "Object Tracking Using Wireless Sensor Networks", Naval Postgraduate School, Monterey, California, September 2005. Available: http://www.scribd.com/

[2] Getting started guide 7430-0022-07, Rev. A, September 2005, Crossbow Technology, Inc.

[3] S. Ergen, "ZigBee/IEEE 802.15.4 Summary", University of California, Berkley, September 2004, Available: pages.cs.wisc.edu

[4] *Wireless medium Access Control (MAC) and Physical layer (PHY) Specification for Low-Rate Wirelwss Personal Area Network* (*WPANs*), IEEE Standard 802.15.4, 2006.

[5] MPR-MIB Users Manual 7430-0021-07, Revision B, June 2006, Crossbow Technology, Inc.

[6] MTS-MDA Sensor Board Users Manual, Revision B, June 2006, Crossbow Technology, Inc.

[7] D. Gay, P. Lewis, E. Brewer, "A Holistic Approach to Network Embedded Systems", University of California, Berkley, 2003. Available: www-csag.ucsd.edu

[8] W. Ye, J. Heidemann, "An Energy-Efficient MAC Protocol for wireless Sensor Networks", University of Southern California, 2005, Available: www.isi.edu

[9] CISCO University Research Program Project: Hierarchical Wireless Security in Low-energy Microsensor Networks, "MultiHop Mesh Networking", San Jose, 2005, Available: http://www.ce.rit.edu/

ABSTRACT

This paper presents the basic concepts of wireless sensor networks and use of embedded software tools through practical exercises. Principles of operating system TinyOS and programming language nesC as embedded software tools that are used for programming applications on wireless sensor and java applications that are used for monitoring the behaviour of a wireless sensor network are briefly given.

DESCRIPTION AND APPLIANCE OF SOFTWARE TOOLS IN WIRELESS SENSOR NETWORK PROGRAMMING

Branislav Novkov, Miladin Jovanović