Algoritam za skladištenje informacija u sistemu za merenje parametara životne sredine u konceptu Industrije 4.0

Ivan Gutai, *Member, IEEE,* Aleksandra Gutai, Marina Subotin, *Member, IEEE*, Platon Sovilj, *Member, IEEE*, Marjan Urekar, *Member, IEEE,* Đorđe Novaković, *Member, IEEE*

***Apstrakt* — U Industriji 4.0 se često dešava da se korisnici uređaja zasipaju nepotrebnim podacima i može se desiti da im promakne nešto što je bitno ili da im se prosledi pogrešan podatak. U merno akvizicionom sistemu u kom se merenje vrši preko DHT22 senzora, tačnosti ± 0.5ºC, nema smisla prikazivati trenutnu temperaturu u obliku 23.46ºC iz dva razloga, koja će biti detaljno opisana u ovom radu. Prvi razlog je što nije metrološki korektno prikazivati korisniku nešto što je suprotno sa specifikacijom proizvođača. Drugi razlog je nepotrebno trošenje prostora u EEPROM-u (Electrically erasable programmable read-only memory , npr. skladištenje celog broja zauzima jedan bajt, dok upotreba broja sa pokretnim zarezom (eng. floating point) zauzima bar četiri bajta. Takođe će biti opisan algoritam koji omogućava zapis podataka 24/7, na svakih 20 minuta u 4 KB EEPROM-a, sa akcentom na robusnosti i metrološkoj ispravnosti skladištenih podataka. Sistem test je vrlo važna disciplina i u ovom radu je odnos pisanja koda i testiranja sistema u srazmeri 1:3.**

***Ključne reči*— Arduino, DHT22, Algoritam, EEPROM, Metrologija, Industrija 4.0, Sistem test, robusnost**

I. Uvod

Moderni softverski inženjer čim čuje pojam skladištenje podataka odmah razmišlja o poslovnim (eng. enterprise) korisnicima, bazama podataka i farmi servera čija cena prelazi nekoliko hiljada eura. Da bi se u jednom merno-informacionom sistemu informacija automatski skladištila, potrebno je malo manje od jednog procenta navedene vrednosti za nabavku hardvera za ovu namenu. Kod otvorenog hardvera sve je nekako dostupno i modularno, pa se sa dodatnim zahtevima i sistem može proširivati. Još jedna motivacija za korišćenje EEPROM-a kao memorije za skladištenje je robusnost, kako proizvođač navodi [1], to je milion ciklusa upisivanja. Pored toga što je hardver robustan, u radu je opisano nekoliko testova celog sistema koji su omogućili da i firmver kojim se omogućava zapisivanje vrednosti u EEPROM, ni u jednoj situaciji ne prepiše izmerenu vrednost u toku istog dana. Takođe, testiran je i scenario kada dođe do prekida napajanja, a rezultat je da u memoriji samo ne postoje vrednosti iz perioda u kom nije bilo napajanja. Uređaj se napaja baterijski sa 5 V, a RTC (Real Time Clock) ima svoju CR2032 bateriju od 3V .

Ivan Gutai – Fakultet tehničkih nauka, Novi Sad, Srbija (e-mail: gutai@uns.ac.rs).

Aleksandra Gutai – Novi Sad, Srbija (e-mail: aleksandra.gutai@gmail.com).

Marina Subotin – Fakultet tehničkih nauka, Novi Sad, Srbija (e-mail: marina.bulat@uns.ac.rs).

Platon Sovilj – Fakultet tehničkih nauka, Novi Sad, Srbija (e-mail: platon@uns.ac.rs).

Marjan Urekar – Fakultet tehničkih nauka, Novi Sad, Srbija (e-mail: marjanurekar@gmail.com).

Đorđe Novaković – Fakultet tehničkih nauka, Novi Sad, Srbija (e-mail: djordjenovakovic@uns.ac.rs).

II. Algoritam

Algoritam se oslanja na funkciju *dayOfTheWeek* iz biblioteke [2]. Svaki dan u nedelji ima svoj broj, a u navedenoj biblioteci dani počinju od nedelje, a brojanje, naravno od nule. Npr. u tom slučaju petak ima broj pet, primenom *dayOfTheWeek* funkcije. Ovakvim brojanjem se memorija od ukupno 4096 bajta, tj. memorijske lokacije od 0 do 4095 mogu izdeliti na sedam plus jedan jednakih delova, sedam za dane u nedelji, a jedan za kontrolnu logiku, što je i prikazano u "(1)".

$$adresa = dan\_u\_nedelji * 504 + dnevni\_brojac * 7 \quad (1)$$

Mapiranje memorije na čipu AT24C32 je prikazano u Tabeli I.

TABELA I
MAPIRANJE MEMORIJE NA ČIPU AT24C32

dan u nedelji	kontrolna logika	oznaka	memorijske lokacije
nedelja	/	0	0-503
ponedeljak	/	1	504-1007
utorak	/	2	1008-1511
sreda	/	3	1512-2015
četvrtak	/	4	2016-2519
petak	/	5	2520-3023
subota	/	6	3024-3527
/	da	/	3528-4095

32 u nazivu sugeriše da čip ima 32 Kb, što je dobro za marketnig, ali često zbunjuje korisnika. Nama je praktičnije da podelimo vrednost sa brojem 8 i da izrazimo tu memoriju kao 4 KB, tj. 4096 memorijskih lokacija. Primenom "(1)", prvi zapis koji će biti napravljen u npr. petak 6.3.2020. u 0:00 je biti zapisan u memorijskoj lokaciji 5*504 +0*7, tj. u memorijskoj lokaciji 2520. Zapis koji će biti napravljen na kraju sedmodnevnog merenja koje je započeto subotom u 0:00 će se nalaziti na memorijskoj lokaciji 6*504 + 71*7, tj. na lokaciji 3521. Vrednosti promenljive *dnevni_brojac* se inkrementuju nakon svakog očitavanja, a mogu biti u opsegu od 0 do 71. U sedam dana se napravi po 72 merenja što je ukupno 504. Svaki od sedam dana ima svoju memorijsku lokaciju za *dnevni_brojac* . Više brojača za istu namenu se koristi samo zato što je jedini slučaj kad se brojač postavlja na

nulu, onaj momenat kada se u jednom danu izvrši 72 merenja. Namerno je izostavljeno vraćanje na nulu u svim ostalim slučajevima, zbog scenarija sa gubitkom napajanja sistema od 5V. Na memorijskim lokacijama između 3528 i 4095 se nalazi kontrolna logika na kojima se nalaze promenljive kao što su *dnevni_brojac* i sve ostale koje omogućavaju funkcionisanje programa. Pored RTC-a, vreme se meri i unutrašnjim 16-bit tajmerom i na svaki sekund se inkrementuje promenljiva *upTime*. Interesanto je što je kompleksnost upotrebe tajmera obrnuto proporcionalna sa bit-nošću, pa su o onim razvojnim sistemima koji ih poseduju, najjednostavniji za korišćenje 32-bit tajmeri. Većina ravojnih sistema ima nekoliko tajmera, npr. Arduino Nano ima tri tajmera, od kojih su dva 8-bit. Na navedeni način se omogućuje praćenje koliko je premena prošlo od uključivanja uređaja (eng. uptime), a zatim se na svaki minut, to vreme i prikazuje. U prototipu uređaja za akviziciju parametara životne sredine, navedeni podatak je čisto informativnog karaktera, dok neki proizvođači hardvera upravo ovakve stvari mogu koristiti za maliciozan kod. Npr. nakon šesnaest uključivanja uređaja i ukupnog uptime-a od petsto devedeset časova promeni se vrednost jednog kontrolnog bita i firmver postanje dosta sporiji ili prestane da radi. Neko od developera to zove greškom (eng. bug), a neko to zove funkcionalnost (eng. feature). Ova mogućnost ostavlja projektantu da omogući nesmetan rad uređaja npr. u garantnom roku ili probnom periodu, a zatim sledi održavanje.

Prilikom svakog merenja se zapisuju četiri vrednosti: izmerena temperatura, izmerena vlažnost vazduha, 32-bitni vremenski žig i izračunata vrednost toplotnog indeksa (eng. heat index). Ukupan broj bajta koji se koristi pri svakom zapisu iznosi sedam. Da je bilo potrebe za dodatnom štednjom memorijskog prostora, ne bi bila upisivana vrednost toplotnog indeksa, koja bi mogla biti izračunata na drugom kraju uz pomoć formule, koja je data u referenciranoj biblioteci. Iako EEPROM podržava milion zapisa, u startu se izbegava višestruko uzastopno zapisivanje. Brži je upis, lakše je pratiti na kojoj memorijskoj lokaciji se nalaze podaci i umesto četiri uzastopna zapisa u ovom slučaju se vrši po jedan zapis za svako merenje.

Ukoliko mi, kao neko ko je sastavio ovaj merno-akvizicioni sistem, proglasimo da je tačnost našeg uređaja ± 1ºC, imamo i više nego dovoljno prostora da koristimo celobrojno (programersko) zaokruživanje. Krajnjem korisniku je omogućeno da letimično pogleda displej i da vidi celobrojnu vrednost, a ne da razaznaje 23.48ºC sobne temperature. Usled tačnosti senzora navedena decimalna informacija je pogrešna, samo zbog viška decimale. Zbog celobrojnog zaokruživanja i deklarisanja da naš uređaj ima tačnost ± 1ºC, sa punim pravom možemo reći da je 23ºC za navedenu vrednost u potpunosti ispravno. Drugi pozitivni ishod je što se zbog kompatibilnosti i uštede prostora celobrojna vrednost smešta u *int8_t* format koji zauzima jedan bajt umesto bar četiri koliko bi zauzeo broj sa pokretnim zarezom. Po jedan bajt se koristi za smeštanje vrednosti izmerene temperature, izmerene vlažnosti vazduha i izračunate vrednosti toplotnog indeksa. Za skladištenje informacije o vremenskom žigu se koristi nešto što se smatra dobrom praksom u programiranju, a to je tzv. „Unix timestamp“ [3]. Iako nije najkraći format zapisa, definitivno je najpraktičniji za dalju obradu. Npr. kada želimo da očitamo vrednosti sa senzora i da ih prikažemo u internet pretraživaču, nema potrebe za bilo kakvom konverzijom. Postoje i rešenja za zapisivanje vremena koje iznose po nekoliko bita, ali uvođenjem takvih „prečica“ znatno se smanjuje period u kom uređaj može samostalno i ispravno da fukncioniše, tj. povećava se učestalost „redovnog održavanja“, što je pre svega i loša programerska praksa.

III. Način pisanja koda

Kod koji čini jedan firmver mora da bude napisan na takav način, da je uredan i jasan. Čak i u nekim slučajevima da bude opisno napisan, da ne mora da sadrži ni jedan komentar. U vremenu kada u većini slučajeva koristimo OOP (objektno orijentisano programiranje), kada nam neko spomene strukturno programiranje, prva asocijacija je tzv. špageti kod. Zaboravljamo da su ljudi koji su napravili OOP zapravo dugi niz godina koristili strukturno programiranje i ukoliko je dobro napisan ne mora da bude objektno orijentisan. Potrebno je da se podsetimo da i dalje možemo puno da uradimo i dosta toga da naučimo iz strukturnog programiranja. Špageti kod je opšte prisutan i pravi se sa copy/paste medodom bez previše razmišljanja. Jedno od važnih pravila je da se ne prave tzv. super funkcije koje rade petnaest stvari istovremeno, nego da se za svaku funkcionalnost izdvaja logika u posebnu funkciju. Nazivi funkcija moraju biti smisleni kao što su *readFromDHT22* ili *readAddressFromEEPROM*. Grupisanje koda se vrši gde je to moguće, a jedan od primera su vrednosti merenja prikazane na slici 1.

```
typedef struct {
  uint32_t timestamp;
  int8_t dhtTemperature;
  int8_t dhtHumidity;
  int8_t dhtCalculatedHeatIndex;
} dhtData;
```

Sl 1. Vrednosti merenja grupisane u strukturi

IV. Sistem test

Testiranje sistema je vrlo bitna disciplina i često se dešava da posle pet sati pisanja firmvera treba petnaest sati da se sve testira, uključujući i granične slučajeve, koji će se desiti npr. u jedan posto slučajeva. Bitna razlika je između testiranja i testiranja sistema je u tome što kod sistem testa složenost raste sa N komponenata, na N i bitno je da možemo da sagledamo širu sliku. Često se desi da pet komponenata zasebno radi u potpunosti ispravno, ali kada se dođe u fazu sastavljanja, kako broj komponenata raste, tako se češće dešavaju nepredviđene stvari. U industriji to ne sme da se dešava i iz tog razloga je bitno da znamo šta se dešava u unutrašnjosti onoga što testiramo, tzv. *white box testing*. Potpuno ista pravila važe i za hardver i za softver, a dobra praksa je da pored projektanta, testira i neko ko taj sistem prvi put vidi, upravo zbog graničnih slučajeva.

Algoritamsko razmišljanje i programerske veštine definitivno omogućavaju da se neki repetitivni poslovi automatizuju. Npr. ukoliko se oslanjamo na izvršavanje nekih aktivnosti u određeno vreme, nećemo provesti 24 h sa papirom i olovkom i testirati šta se kad desilo. Upotrebom bukvalno dve *for* petlje koje se nalaze jedna ispod druge gde prva broji od nula do dvadeseti tri a druga od nula do pedeset

devet, može se simulirati kompletan dan. Prototip modularnog uređaja za akviziciju parametara životne sredine je testiran na više načina. Nakon višednevnog testiranja od po nekoliko sati, sedmodnevni test je započet u sredu 29.1.2020. u 16:20, a završen je 5.2.2020 u 16:20. Takođe, vrednost promenljive *dnevni_brojac* namerno nije vraćena na nulu, već je ostala na vrednosti 65, od ranijih testiranja. Sa ovakvim podešavanjima, vrednost iz memorijske lokacije 1512 je ona koja je nastala u 21:40. Sa merenjem u 23:40 se završava merenje za taj dan. Nakon te vrednosti u memoriji se nalaze one od početka dana, itd. Prilikom pregleda zapisa u memoriji može se primetiti da je na lokaciji 2009 za sredu zapisana vrednost koja je izmerena u 21:20. Na navedeni način je omogućena robusnost, tj. u toku istog dana neće doći do prepisivanja zapisa. Detaljno analiziranje gde je šta zapisano je bilo potrebno samo zbog testiranja robusnosti, a krajnje korisnika se to ne tiče. Podaci su spremni da se šalju sa sistema u JSON (JavaScript Object Notation) formatu, što na drugom kraju omogućava olakšano parsiranje. Postoji niz mogućnosti kako poslati ove podatke dalje u sistem, a dva najlogičnija su slanje preko serijskog porta, a zatim i dodavanjem Wi-Fi adaptera u sistem. Popularna komponenta koja dodaje Wi-Fi na Arduino razvojne sisteme je ESP-07 proizvođača AiThinker [4], koji je zasnovan na ESP8266 [5] čipu.

Kontrola vremena

U prototipu modularnog uređaja za akviziciju parametara životne sredine, kontrola vremena se vrši na dva načina. Prvi je RTC, a drugi je tajmer. Koristi se samo jedan tajmer, iako postoji mogućnost za istovremenu upotrebu više. Matematička operacija moduo „%“ se koristi da se odredi kada je 20, 40 ili 60 minut svakog sata. Proverava se kada je moduo od 20 jednak 0. Ove vrednosti su birane zbog preglednosti, a u sistemu koji npr. ne bi imao tačno vreme, moglo je da se sa tajmerom broje sekunde i da se računa relativno vreme u odnosu na to kada je sistem dobio napajanje. Često se spominje i upotreba RTOS-a (Real Time Operating System) kao vida za paralelno izvršavanje nekoliko zadataka, iako ta potreba u navedenom scenariju zaista ne postoji. Npr. ukoliko je potrebno da se prati nešto na 3, 5 i 19 minuta, ukoliko to ne može da se reši upotrebom različitih provera, uvek postoji opcija da se oslonimo na ostale tajmere koje razvojni sistem poseduje, a najpopularniji imaju između 3 i 9. Kao što se upotrebom 16-bitnog tajmera broj 1 na Arduino Nano računa koliko je minuta prošlo od pokretanja sistema, druga dva tajmera, tajmere 0 i 2 možemo upotrebiti za neki vid paralelnog izvršavanja. Koliko god se trudimo da pratimo trendove, uvek moramo voditi računa o tome da ne komplikujemo život sebi, ali ni korisnicima naših uređaja. Još jedna stvar o kojoj treba voditi računa je upravljanje vremenom potrebnim za razvoj. Nema odlaganja i čekanja da uređaj bude savršen i nagomilavanja nepotrebnih funkcionalnosti. Korisnik je nestrpljiv i zahtevan i očekuje da uređaj radi ispravno. Iz navedenih razloga dobra praksa je da se naprave iteracije koje mogu minimalno trajati dve nedelje i kako faze projektovanja odmiču, tako se uključuje i korisnik kojem je lakše da sagleda funkcionalnost uređaja. Iako je većina ovakvih uređaja projektovana da radi samostalno, ljudski fakor je tu najbitniji kod testiranja.

Zaključak

Sada je vreme kada marketing službe mnogo češće pišu tehničke specifikacije uređaja u odnosu na inženjere i često možemo pomisliti da je taj i taj proizvod neke kompanije baš onaj koji nam je potreban. Kod otvorenog hardvera postoji niz naprednijih i modernijih razvojnih sistema čija je cena slična uređaju Arduino Nano, ali po jednoj stvari ne mogu ni približno da se porede. Broj primera za „starije“ i oprobane razvojne sisteme je dosta veći, pa se iz velike mase može izdvojiti nekoliko kvalitetnih. Uređaj većinu stvari radi samostalno, ali moramo voditi računa da je i ljudski faktor vrlo bitan u industriji. U verziji 0.1 prototipa modularnog uređaja za akviziciju parametara životne sredine, svi podaci se prikupljaju automatski, a jednom nedeljno je potrebno da ih čovek preuzme, pošto će zbog namerno ostavljene manje količine memorije doći do prepisivanja na nedeljnom nivou.

Zahvalnica

Ovaj rad je delom podržan od strane projekta ELEMEND (šifra projekta: 585681-EEP-1-2017-EL-EPPKA2-CBHE-JP).

Literatura

[1] https://ww1.microchip.com/downloads/en/DeviceDoc/doc0336.pdf
[2] https://github.com/adafruit/RTClib
[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date
[4] http://wiki.ai-thinker.com/_media/esp8266/a018ps01.pdf
[5] https://www.espressif.com/en/products/hardware/esp8266ex/overview

Abstract

In Industry 4.0 it is very common that users are overwhelmed with unnecessary data and sometimes they can miss something relevant or bad data can be forwarded. In a measuring-acquisition system that uses a DHT22 sensor, which has an accuracy of ± 0.5ºC, there is no point for showing temperature as 23.46ºC for two reasons, which would be described in this paper. The first reason is metrological incorrectness for showing to end-user something that is not in compliance with manufacturer specifications. The second reason is the unnecessary wasting of space in EEPROM. For example, storing integer value takes 1 byte and storing floating-point value takes 4 bytes. The algorithm which uses 4 KB of EEPROM as a storage for data acquisition every 20 minutes, 24 hours a day, 7 days a week will be described. The accent is on robustness and metrological validity for storing data. The system test is a very important discipline, and in this paper, writing code and system tests have a 1:3 ratio.

The algorithm for storing measured environmental parameters in EEPROM in the Concept of Industry 4.0

Ivan Gutai, Aleksandra Gutai, Marina Subotin,
Platon Sovilj, Marjan Urekar, Đorđe Novaković