## LABORATORIJSKA VEŽBA BR. 3

# Potrošnja električne energije

CILJ VEŽBE

- Upoznavanje sa karakteristikama jedinice za napajanje
- Upoznavanje sa načinima za smanjivanje elektične potrošnje
- Upoznavanje sa vrstama baterija i njihivim karakteristikama
- Upoznavanje sa modovima rada CPU jedinice i ostalih komponenta senzorskog čvora
- Testiranje softverskih metoda – korišćenje biblioteke Low-Power
- Testiranje optimizacije potrošnje mikrokontrolera
- Testiranje optimizacije potrošnje senzora
- Upoređivanje dobijenih podataka
- Donošenje zaključaka na osnovu urađenih testiranja

POTREBNA OPREMA

- Računar sa instaliranim razvojnim softverom za Arduino
- Arduino UNO + Mega 3 komplet
- DHT i BH1750 senzore
- Multimetar
- Komplet alata za montažu

## TEORIJSKE OSNOVE

Jedan od bitnijih faktora u dizajnu BSM-a, po kome se BSM-e razlikuju od ostalih bežičnih mreža, odnosi se na potrošnju električne energije kao i efikasno upravljanje tom potrošnjom. Bežični senzorski čvorovi predstavljaju potpuno autonomne uređaje ispred kojih stoji zadatak da se samostalno organizuju tj. formiraju određenu mrežnu topologiju, da prikupe podatke o nekoj pojavi iz prirode i da te podatke preusmere prema glavnom senzorskom čvoru (*sink*). Jedinica za napajanje predstavlja ključni deo svakog SČ jer od njenog funkcionisanja zavisi rad svih ostalih komponenti SČ. Najčešće se sastoji od baterijskog izvora jako ograničenog kapaciteta(<0.5 Ah, 1.2 V), DC-DC pretvarača kao i alternativnih kola koja omogućuju korišćenje energije iz prirode. Svi delovi SČ se napajaju iz ove jedinice pa se, zbog ograničenog kapaciteta koja ona ima, kao osnovni zahtev postavlja da rad svih jedinica u SČ bude energetski efikasan. Kako je u većini slučajeva zamena baterije jako otežana, čak nemoguća zbog nepristupačnih terena i velikog broja SČ-ova, jasno je da je životni vek SČ, a samim tim i cele aplikacije u BSM, direktno zavistan od količine energije koja je skladištena u bateriji. U principu postoje dva aspekta u razvoju ove jedinice kako bi se omogućio maksimalni životni vek SČ. Prvi je da se postigne što veća gustina skladištenja energije koja se smešta u bateriju i drugi je da se omogući dopunjavanje ili korišćenje energije iz prirodnih izvora (*energy scavenging unit*).

U okviru ove vežbe student treba da nauči na koji način može efikasno da smanji potrošnju bežičnog senzorskog čvora sa ciljem da omogući što duži životni vek aplikacije u BSM. Student će se upoznati sa vrstama baterijskog napajanja, modovima rada CPU jedinice kao i ostalim mehanizmima koji se mogu

primeniti a sve u cilju uštede električne energije (smanjivanje frekvencije, smanjivanje radnog napona, isključivanje pojedinih delova, …)

## Baterija

Zadnjh godina razvijen je veliki broj različitih baterija koje su namenjene uređajima koji mogu bežično da komuniciraju. Po kvalitetu i rasprostranjenosti izdvojila su se tri tipa tih baterija: NiMh – Nikl metal hibridne baterije (*Nickel–metal hydride*), Li-Ion – Litijum jonske baterije (*Lithium-ion*) i Li polimer – Litijum polimer baterije (*Lithium-ion polymer*). Svaki od ovih tipova baterija ima jedinstvene karakteristike koje odgovaraju ili ne odgovaraju za primenu u BSM. Poznavanje specifičnosti ovih baterija u pogledu njihovog nominalnog napona, kapaciteta, energetske gustine, specifične snage, vremena punjenja i vremena pražnjenja predstavlja prvi korak u izboru odgovarajućeg rešenja za napajanje SČ.

Posmatrano sa sistemske strane jedna dobra baterija treba da zadovolji sledeće uslove:
1) veliku energetsku gustinu (vidi Tabelu 1);
2) veliku specifična energiju;
3) mali napon po ćeliji (0.5-1.0 V);
4) efikasno konfigurisanje ćelija baterije, kako bi se izbegla potreba za DC/DC konvertorima;
5) mogućnost dopunjavanja baterija.

**Tabela 1.** Energetska gustina kod primarnih i sekundarnih baterija

| | ***Energija [J/cm$^3$]*** | | | | |
|---|---|---|---|---|---|
| ***Tip baterije*** | Zinc-air | Lithium | Alkaline | NiMHd | NiCd |
| ***Primarna*** | 3780 | 2880 | 1200 | | |
| ***Sekundarna*** | | 1080 | | 860 | 650 |

Kapacitet baterije se prikazuje u miliampersatima (*mAh*), a to nam u suštini govori koliko sati baterija može kontinuirano da daje struju navede jačine u mA. Na primer, baterija od *600 mAh* teoretski može da daje *600 mA* u periodu od jednog sata. Na osnovu ovoga ako znamo prosečnu potrošnju našeg senzorskog čvora možemo vrlo lako proceniti prosečno vreme trajanja našeg uređaja korišćenjem sledeće jednačine:

**Kapacitet baterije (mAh) / Prosečna potrošnja struje (mA) = Prosečno trajanje života (h)**

Primer: ako je prosečna potrošnja senzora **150mA** a mi imamo bateriju koja ima kapacitet od **600mAh** jednostavnom računicom dolazimo do sledećeg rezultata:

600 mAh / 150 mA = 4 h.

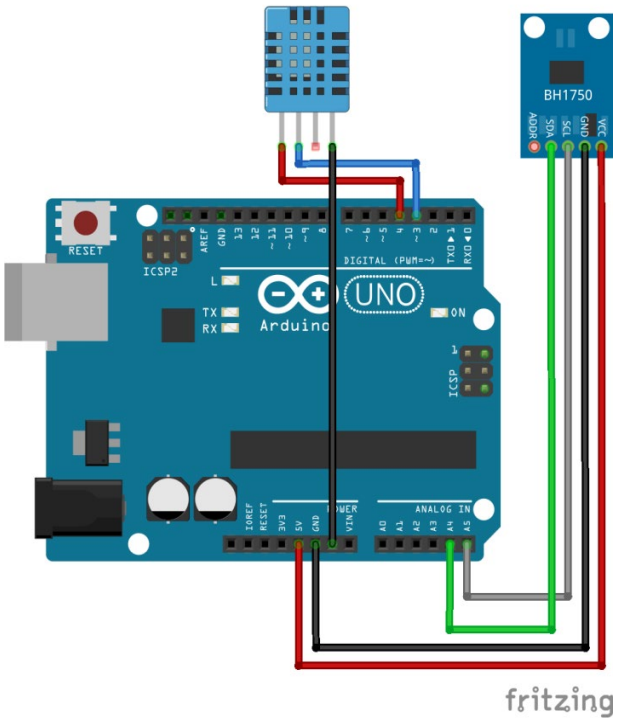
Međutim, ukoliko želimo da naš senzorski uređaj ima znatno veću autonomiju rada (1 godinu), gore navedene pretpostavke za prosečnu potrošnju kao i korišćeni kapacitet baterije nam ne idu u prilog. Najjednostavniji način za postizanje veće autonomije rada senzorskog uređaja bi bio da se poveća kapacitet baterije. Međutim, da bismo to realizovali, trebala bi nam baterija od otprilike 1500 Ah (kao poređenje navedimo da akumulatori u automobilu imaju kapacitet od oko 120 Ah), što ergonomski nije moguće. Prema tome, ništa nam drugo ne preostaje već da pokušamo da iskoristimo sve raspoložive metode kako bi potrošnju energije u senzorskom čvoru smanjili na najmanju moguću meru.

Intenzitet rada senzorskog čvora u jednoj aplikaciji za BSM se veoma razlikuje. Većinu svog vremena on ne radi ništa da bi se u trenutku detekcije nekog događaja on aktivirao i u tom trenutku u celoj mreži imamo jako intezuvan saobraćaj. Samim tim, ako se u periodu neaktivnosti senzorskog čvora ne generišu nikakvi podaci o premenama u prirodi i ne šalju podaci preko radio kanala, tada je senzorski čvor u osnovi neaktivan, te je potrebno u tom periodu maksimalno smanjiti njegovu potrošnju. Takođe, ako želimo smanjiti ukupnu veličinu senzorskog uređaja, potrebno je voditi računa o veličini, a samim time i kapacitetu baterije. Kao primer uzećemo senzorski uređaj čiji kapacitet baterije ne prelazi otprilike 3000 mAh. Ukoliko želimo osigurati autonomiju rada od godinu dana, tada nije teško izračunati da bi potrošnja uređaja trebala biti u nivou od nekoliko uA u periodu njegove neaktivnosti.

Na linku http://oregonembedded.com/batterycalc.htm možemo izvršiti procenu trajanja života određene baterije ako unesemo odgovarajuće parametre o potrošnji u periodu aktivnosti i neaktivnosti uređaja kao I vremenskom trajanju aktivnog perioda.

## ZADATAK

***Student treba da izvršiti procenu koliko će biti životni vek jedne aplikacije u BSM na primeru vežbe iz prošlog časa (očitavanje temperature i vlažnosti) ako se koristi baterija od 3000 mAh i to bez korišćenja metoda uštede energije i uz korišćenje metoda uštede koje su navedene u ovoj vežbi. U primeru sa prošle vježbe, cilj je napraviti takvu senzorsku mrežu koja će meriti temperaturu/ vlagu i osvetljenje u period od 1 minuta, dok će ostatak vremena između dva očitanja biti neaktivna.Pri tome treba koristiti*** *DHT11/22* ***senzor za očitanje temperature/vlage i prema mogućnostima senzor*** *BH1750* ***za očitanje nivoa osvetljenja****.*




## Postupak rešavanja zadatka:

Da bi smo mogli da utičemo na smanjivanje potrošnje električne energije prvo je potrebno da se upoznamo sa najvećim potrošačima baterije. To su pre svega komponente senzorskog čvora: računarska jedinica, jedinica za komunikaciju i senzorska jedinica. Redom ćemo pokazati na primerima kako jednostavnim intervencijama možemo smanjiti potrošnju senzorskog čvora.

### 1. Ušteda električne energije u računarskoj jedinici

Ubacite u vaš Arduino Uno R3 sledeći kod i proverite kolika je potrošnja Arduina:

```
void setup () { }
void loop () { }
```

Nakon toga pustite da se izvršava sledeći kod:

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include <avr/pgmspace.h>
#define DHTPIN 3
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
BH1750 lightMeter(0x23);
void readTempHum();
void readLight();

void setup() {
pinMode(4,OUTPUT);
digitalWrite(4,LOW);
Serial.begin(9600);
Serial.println(F("DHTxx test!"));
```

```
dht.begin();
lightMeter.begin(BH1750_CONTINUOUS_HIGH_RES_MODE);
Serial.println(F("BH1750 Test"));
}

void loop() {
readTempHum();
readLight();
}

void readTempHum() {
digitalWrite(4, HIGH);
// Wait at least 2 seconds seconds between measurements.
unsigned long previousMillis = millis();
unsigned long currentMillis = millis();
while (currentMillis - previousMillis <= 2300) {
// save the last time you read the sensor
currentMillis = millis();
}
Serial.print(F("Humidity: "));
Serial.print(dht.readHumidity());
Serial.print(F(" %\t"));
Serial.print(F("Temperature: "));
Serial.print(dht.readTemperature());
Serial.println(F(" *C "));
}

void readLight() {
Serial.print(F("Light: "));
Serial.print(lightMeter.readLightLevel());
Serial.println(F(" lx"));
}
```

**NAPOMENA:** potrebno je instalirati **Low-Power** biblioteku od ***Rocketscream***-a:

```
#include <LowPower.h>

void setup() {}

void loop() { LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF); }
```

Potrebno je uporediti dobijene rezultate i videti kolika je razlika između potrošnje Arduino sistema u ova dva slučaja.

## 2. Optimizacija potrošnje mikrokontrolera

Na slici se nalazi tablica u kojoj možemo da vidimo kolika je potrošnja **ATmega328P** procesora (koji se nalazi u Arduino Uno R3) u zavisnosti od izabranog moda rada procesora. Uočite da različiti modovi sa smanjenom potrošnjom energije onemogućavaju razne interne satove i komponente. Takođe, veoma je važno da se vidi na koji način možemo da ‘’probudimo’’ procesor iz stanja mirovanja, tj. da li je to moguće putem **interrupta** ili **timer-a**. Na sledećem linku možemo pronaći detaljnije objašnjenje koje nam govori koje komponente procesora možemo ugasiti da bi maksimalno smanjili potrošnju procesora:

http://www.gammon.com.au/power

**Table 9-1.** Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

| | Active Clock Domains | | | | | Oscillators | | Wake-up Sources | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Sleep Mode** | $clk_{CPU}$ | $clk_{FLASH}$ | $clk_{IO}$ | $clk_{ADC}$ | $clk_{ASY}$ | Main Clock Source Enabled | Timer Oscillator Enabled | INT1, INT0 and Pin Change | TWI Address Match | Timer2 | SPM/EEPROM Ready | ADC | WDT | Other I/O | Software BOD Disable |
| Idle | | | X | X | X | X | X(2) | X | X | X | X | X | X | X | |
| ADC Noise Reduction | | | | X | X | X | X(2) | X(3) | X | X(2) | X | X | X | | |
| Power-down | | | | | | | | X(3) | X | | | | X | | X |
| Power-save | | | | | X | | X(2) | X(3) | X | X | | | X | | X |
| Standby(1) | | | | | | X | | X(3) | X | | | | X | | X |
| Extended Standby | | | | | X(2) | X | X(2) | X(3) | X | X | | | X | | X |

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Za potrebe izrade današnje vežbe izabraćemo najnižu potrošnju energije u stanju mirovanja, tzv. *power-down mode*. Da bismo to uradili potrebno je da uključimo *Lightweight low-Power* biblioteku za Arduino od Rocketscream-a koja podržava *power-down mode* pa je potrebno da je instaliramo u PlatformIO. (Otkucajte: **platformio lib search Low-Power** za traženje biblioteke).

Kao što je detaljnije navedeno na sajtu http://www.gammon.com.au/power, najveći potrošač energije pored izbora da processor radi u power-down mode je i omogućen ADC (*Analog to Digital Conversion*) te *brown-out* (BOD) funkcionalnost/detekcija. Jednostavnim pozivom naredbe u Arduinu isključićemo ADC i BOD i izabrati power-down mode za rad procesora.

**LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF)**

U ovom načinu rada gotovo sve komponente procesora su onemogućene, dok samo spolnji *interrupt* ili *timer* (kao što je watchdog timer ili *Real Time Clock - RTC*) mogu probuditi procesor. U sklopu današnje vežbe mi ćemo koristiti *watchdog timer* za buđenje iz *power-down sleep* stanja. *Watchdog timer* je komponenta koja se nalazi unutar AVR procesora koji resetuje procesor ako se otkrije da je zaustavljena normalna egzekucija programa. Obično se *watchdog* koristi da bi se nepouzdani program ili program sa dosta grešaka izvodio sa više stabilnosti. Međutim, moguće je koristiti *watchdog timer* za buđenje procesora iz *power-down sleep* stanja, umesto da ga koristite za resetovanje.

**Table 10-2.** Watchdog Timer Prescale Select

| WDP3 | WDP2 | WDP1 | WDP0 | Number of WDT Oscillator Cycles | Typical Time-out at $V_{CC}$ = 5.0V |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 2K (2048) cycles | 16 ms |
| 0 | 0 | 0 | 1 | 4K (4096) cycles | 32 ms |
| 0 | 0 | 1 | 0 | 8K (8192) cycles | 64 ms |
| 0 | 0 | 1 | 1 | 16K (16384) cycles | 0.125 s |
| 0 | 1 | 0 | 0 | 32K (32768) cycles | 0.25 s |
| 0 | 1 | 0 | 1 | 64K (65536) cycles | 0.5 s |
| 0 | 1 | 1 | 0 | 128K (131072) cycles | 1.0 s |
| 0 | 1 | 1 | 1 | 256K (262144) cycles | 2.0 s |
| 1 | 0 | 0 | 0 | 512K (524288) cycles | 4.0 s |
| 1 | 0 | 0 | 1 | 1024K (1048576) cycles | 8.0 s |
| 1 | 0 | 1 | 0 | Reserved | |
| 1 | 0 | 1 | 1 | | |
| 1 | 1 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | | |
| 1 | 1 | 1 | 1 | | |

Na gornjoj slici možemo videti da postoje različiti periodi unutar kojih *watchdog timer* može probuditi naš procesor. Ako želimo što više držati procesor u *power-down modu*, tada on preko watchdog timer-a može maksimalno biti 8 s. Jednostavnom modifikacijom gore navedene naredbe postavljamo procesor u power-down mode na period od 8 sekundi:

```
LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF)
```

Vaš zadatak je da modifikujete navedenu skriptu datu u vežbi na način da joj omogućite korišćenje gore navedenih preporuka za očitavanje senzora DHT11/22 i BH1750 senzora i to tako što ćete očitavati senzor svake minute (period očitavanja iznosi 1 min.) i to uz smanjenu potrošnju energije.

## 3. Optimizacija potrošnje senzora

Da bi na nivou senzorske jedinice mogli da utičemo na smanjivanje potrošnje potrebno je da se bolje upoznamo sa datasheet korišćenih senzora. Ako pogledamo datasheet za senzor DHT11/22 videćemo da on u tzv. idle mode-u (kada ne radi očitanja) troši otprilike **50µA**, što je ipak previše za naš senzorski uređaj. Najjednostavniji način da smanjimo potrošnju našeg senzora DHT11/22 bio bi da ga isključimo sa napajanja u periodu njegove neaktivnosti. To možemo da uradimo jednostavno tako da se senzor napaja sa jednog od digitalnih pin-ova u periodu aktivnosti, te u periodu neaktivnosti softverski isključimo taj digitalni pin i na taj način mu uskratimo napajanje. Takođe, u periodu aktivnosti potrošnja takvog senzora je otprilike **1,5 mA**. Kod BH1750 senzora potrošnja u periodu neaktivnosti je **1.0µA** što je čak povoljno za naš scenario pa ga nećemo spajati na digitalni pin. Prema slici spojite senzore i testirajte kod koji je dat u prilogu ove vežbe. Primetite da se kod razlikuje jedino u tome što pre početka merenja palimo digitalni pin sledećem naredbom:

```
digitalWrite(4, HIGH);
```

## 4. Ostali vidovi za smanjivanje potrošnje

U nastavku slede još neka uputstva koja dodatno mogu smanjiti ukupnu potrošnju senzorskog čvora, a koje nažalost nećemo u sklopu vežbe pokriti:

- Pokretanje procesora na nižoj frekvenciji
- Korišćenje manjeg napona za rad procesora
- Isključite nepotrebne interne module u software-u (npr. SPI, I2C, serijski, ADC)
- Isključite otkrivanje redukcije struje (BOD)
- Isključite analogno-digitalne pretvarač (ADC)
- Isključite watchdog timer
- Stavite procesor u režim smanjene potrošnje
- Ne koristite regulatore napona koji vam ne trebaju
- Nemojte koristiti indikatore/ekrane (LED indikator, LCD ekran)
- Probudite procesor iz sna samo kada je to potrebno
- Isključivanje (s MOSFET-om) vanjskih uređaja (npr. SD kartice, senzori temperature)

Programski kod sa uključenim uštedama za očitavanje senzora:

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include <avr/pgmspace.h>
#include <LowPower.h>
#define DHTPIN 3
#define DHTPOWERPIN 4
#define BHPOWERPIN 5
#define DHTTYPE DHT22
```

```
DHT dht(DHTPIN, DHTTYPE);
BH1750 lightMeter(0x23);
void readTempHum();
void readLight();

void setup() {
pinMode(DHTPOWERPIN,OUTPUT);
digitalWrite(DHTPOWERPIN,HIGH);
pinMode(BHPOWERPIN,OUTPUT);
digitalWrite(BHPOWERPIN,HIGH);
delay(1000); // give some time to send data over Serial before going to sleep

Serial.begin(115200);
Serial.println(F("DHTxx test!"));
dht.begin();
lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE);
Serial.println(F("BH1750 Test"));
delay(100); // give some time to send data over Serial before going to sleep
}

void loop() {
// Enter power down state for 8 s with ADC and BOD module disabled
LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
delay(100); // give him some time to wake up from sleep :D
readLight();
readTempHum();
delay(100); // give some time to send data over Serial
}

void readTempHum() {
digitalWrite(DHTPOWERPIN, HIGH);
// Wait at least 2 seconds seconds between measurements.
unsigned long previousMillis = millis();
unsigned long currentMillis = millis();
while (currentMillis - previousMillis <= 2300) {
// save the last time you read the sensor
currentMillis = millis();
}

Serial.print(F("Humidity: "));
Serial.print(dht.readHumidity());
Serial.print(F(" %\t"));
Serial.print(F("Temperature: "));
Serial.print(dht.readTemperature());
Serial.println(F(" *C "));
}

void readLight() {
digitalWrite(BHPOWERPIN,HIGH);
```

```
Serial.print(F("Light: "));
Serial.print(lightMeter.readLightLevel());
Serial.println(F(" lx"));
}
```

## ZAKLJUČAK

Korišćenjem navedenih uputstva moguće je veoma jednostavno optimizirati potrošnju Arduino uređaja. Najveći potrošači uz mikrokontroler su LED diode, regulatori napona, ADC i BOD. Čak je i watchdog timer ponekad potrebno isključiti i zameniti ga sa preciznijim satovima, kao što je RTC (npr. DS3232), čija je potrošnja znano manja od watchdog-a. Uz to, moguće je koristiti i Arduino uređaje koji rade na manjem naponu (Arduino Pro Mini verzija 3V3), pa regulatori napona u tom slučaju nisu ni potrebni. Postoje i verzije Arduina koje rade i na manjoj frekvenciji (upravo spomenuti Arduino Pro Mini na 3V3 radi na frekvenciji od 8 MHz) pa i time dodatno dobijamo na kapacitetu baterije.

**Korisni linkovi:**


[1] http://oregonembedded.com/batterycalc.htm
[2] http://www.microchip.com/forums/m672837.aspx
[3] https://github.com/rocketscream/Low-Power
[4] http://www.gammon.com.au/power
[5] http://www.home-automation-community.com/arduino-low-power-how-to-run-atmega328p-for-a-year-on-coin-cell-battery/
[6] http://www.elechouse.com/elechouse/images/product/Digital%20light%20Sensor/bh1750fvi-e.pdf
[7] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
[8] https://learn.adafruit.com/low-power-wifi-datalogging/overview
[9] https://edwardmallon.wordpress.com/2014/05/21/using-a-cheap-3-ds3231-rtc-at24c32-eeprom-from-ebay/
[10] https://www.youtube.com/watch?v=urLSDi7SD8M
[11] https://www.youtube.com/watch?v=iMC6eG24S9g