**SVEUČILIŠTE U SPLITU**

**FAKULTET ELEKTROTEHNIKE STROJARSTVA I BRODOGRADNJE**

**SEMINARSKI RAD IZ KOLEGIJA MIKROREGULATORI I UGRADBENI MREŽNI SUSTAVI**

**NAČINI UŠTEDE ENERGIJE MIKROKONTROLERA ARDUINO UNO**

**IVAN SIČAJA**

Split, siječanj 2021.

# SADRŽAJ

[1. UVOD 3](#_Toc61795823)

[2. GRAĐA MIKROKONTROLERA ARDUINO UNO 4](#_Toc61795824)

[3.METODE UŠTEDE ENERGIJE 6](#_Toc61795825)

[3.1 UŠTEDA ENERGIJE SMANJENJEM FREKVENCIJE RADA MIKROKONTROLERA 6](#_Toc61795826)

[3.2 KORIŠTENJE „DEEP SLEEP“ MODOVA S INTERUPTIMA 7](#_Toc61795827)

[3.3 ELEKTRIČNA SHEMA SPAJANJA 8](#_Toc61795828)

[4.TEST 9](#_Toc61795829)

[5. ZAKLJUČAK 10](#_Toc61795830)

[6. PROGRAMSKI KOD 11](#_Toc61795831)

[7. REFERENCE 14](#_Toc61795832)

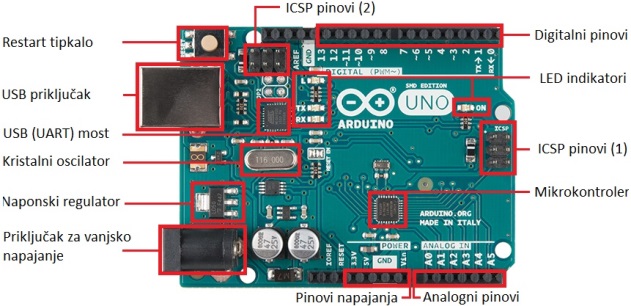
# UVOD

U ovom seminarskom radu ukratko ćemo objasniti načine na koje možemo uvelike umanjiti potrošnju električne energije mikrokontrolera, za naš slučaj koristit ćemo mikrokontroler Adruino Uno. Sve će biti potkrijepljeno programskim kodom te konačnim zaključkom I komentarima vezanim za projekt.

# GRAĐA MIKROKONTROLERA ARDUINO UNO

Ušteda potrošnje električne energije se zasniva na tome da za svako povećanje uštede električne energije dolazi do smanjenja funkcionalnosti određenog mikrokontrolera odnosno dolazi do isključenja rada određenog fizičkog sklopovlja koje se nalazi ili pak brzine rada određenog sklopovlja koje se nalazi na samom mikrokontroleru.

Građa mikrokontrolera Arduino Uno prikazana je na sljedećoj slici.



Slika 2.1. Građa Arduino Uno mikrokontrolera

Iz slike 2.1 vidljivo je da su osim pinova, na mikrokontroleru prisutne i još neke komponente koje ćemo pobrojati u nastavku. To su redom;

- mikrokontroler ATmega328

-komponente za vanjsko napajanje

- restart tipkalo

- naponski regulator

-kristalni oscilator

- čip za serijsku komunikaciju (UART most)

- USB tipa B (eng. Universal Serial Bus)

- LED (eng. Light Emitting Diode)

# 3. METODE UŠTEDE ENERGIJE

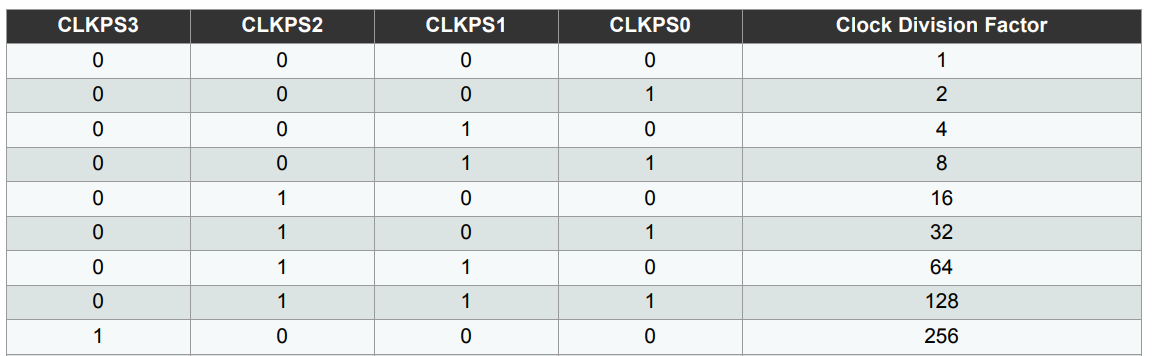
Prva stvar na koju ćemo se fokusirati je kristalni oscilator, potom ćemo se fokusirat ne različite „Deep sleep“ načine rada mikrokontrolera

## 3.1 UŠTEDA ENERGIJE SMANJENJEM FREKVENCIJE RADA MIKROKONTROLERA

Kristalni oscilator predstavlja titrajni izvor frekvencije 16 MHz koji služi kao izvor taktnog signala čija se frekvencija može prilagođavati ovisno o našim potrebama. Prilagođavanje frekvencije koja nam je potrebna vrši se u određenim registrima, shodno tomu dobivene frekvencije mogu biti samo frekvencije dobivene dijeljenjem osnovne frekvencije djeliteljem većim od 1. Također ovo je komponenta bez koje bi rad mikrokontrolera bio nemoguć jer sam mikrokontroler ne bi imao nikakvu predodžbu o vremenu, što bi za sobom povuklo vrlo ozbiljne posljedice, to jest ne bi znali da li neki proces kasni i koliko kasni, da li je to kašnjenje beznačajno ili ne, što bi izazvalo ogromnu pomutnju u radu mikroprocesora koji sam po sebi zahtjeva vrlo stroge i precizne naredbe kako bi obradio jako veliki broj informacija u što kraćem vremenskom periodu što je odlika suvremenih mikroprocesora.

Budući da je ovo sklop koji određuje brzinu rada mikrokontrolera, smanjivanjem takta frekvencijskog oscilatora smanjujemo potrošnju mikrkontrolera. Promjena frekvencije rada mikrokontrolera naziva se preskaliranje. Kako što je vidljivo na sljedćoj slici moguće je dijeljenje osnovne frekvencije s 1,2,,4,8,16,32,64,128 I 256 puta. S kojim brojem želimo dijeliti osnovnu frekvenciju određujemo upisom heksa-decimalnog zapisa određenog reda u CLKPR registar.

Npr. Upisom “CLKPR=0x01” frekvenciju dijelimo s brojem 2 odnosno naš mikrokontroler radi frekvencijom 16MHz/2 odnosno s 8Mhz, upisom “CLKPR=0x02” frekvenciju dijelimo s brojem 4 odnosno naš mikrokontroler radi frekvencijom 16MHz/4 odnosno s 4Mhz.



Slika 3.1. Frekvencije preskaliranja s određenim registrima [1]

## 3.2 KORIŠTENJE „DEEP SLEEP“ MODOVA S INTERUPTIMA

„Deep sleep“ modovi koji su dostupni na mikroprocesoru ATmega328P, koji mi koristimo su:

-SLEEP\_MODE\_IDLE – omogućava najmanju uštedu

-SLEEP\_MODE\_ADC

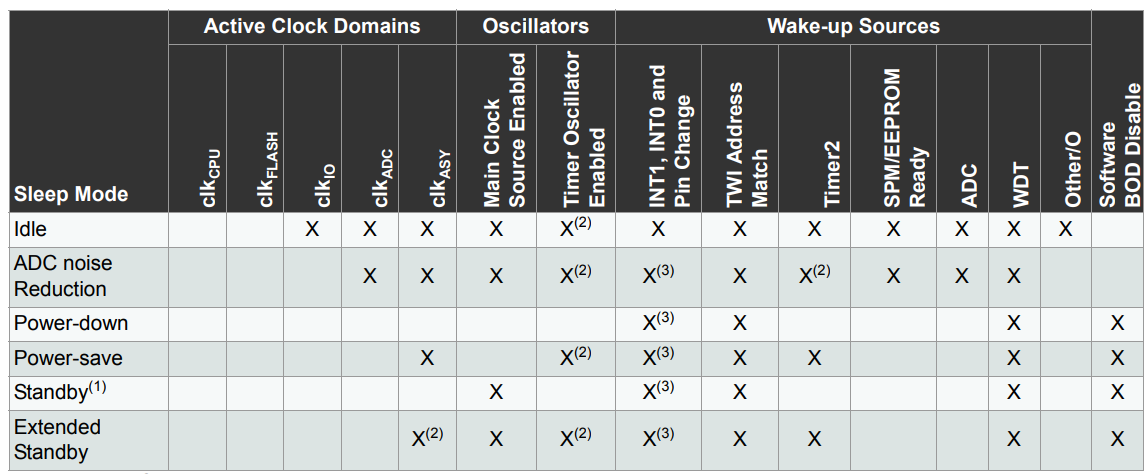
-SLEEP\_MODE\_PWR\_SAVE

-SLEEP\_MODE\_STANDBY

-SLEEP\_MODE\_EXTENDED STANDBY

-SLEEP\_MODE\_PWR\_DOWN – omogućava najveću uštedu

Korištenje sklopovlja prema kojem se razlikuju navedeni modovi uštede energije prikazani su na sljedećoj slici. Ove i sve ostale bitne informacije o određenom elektroničkoj stavci možete pronaći u „datasheet“ određenog proizvoda, u našem slučaju mikrokontrolera ATMega328P.



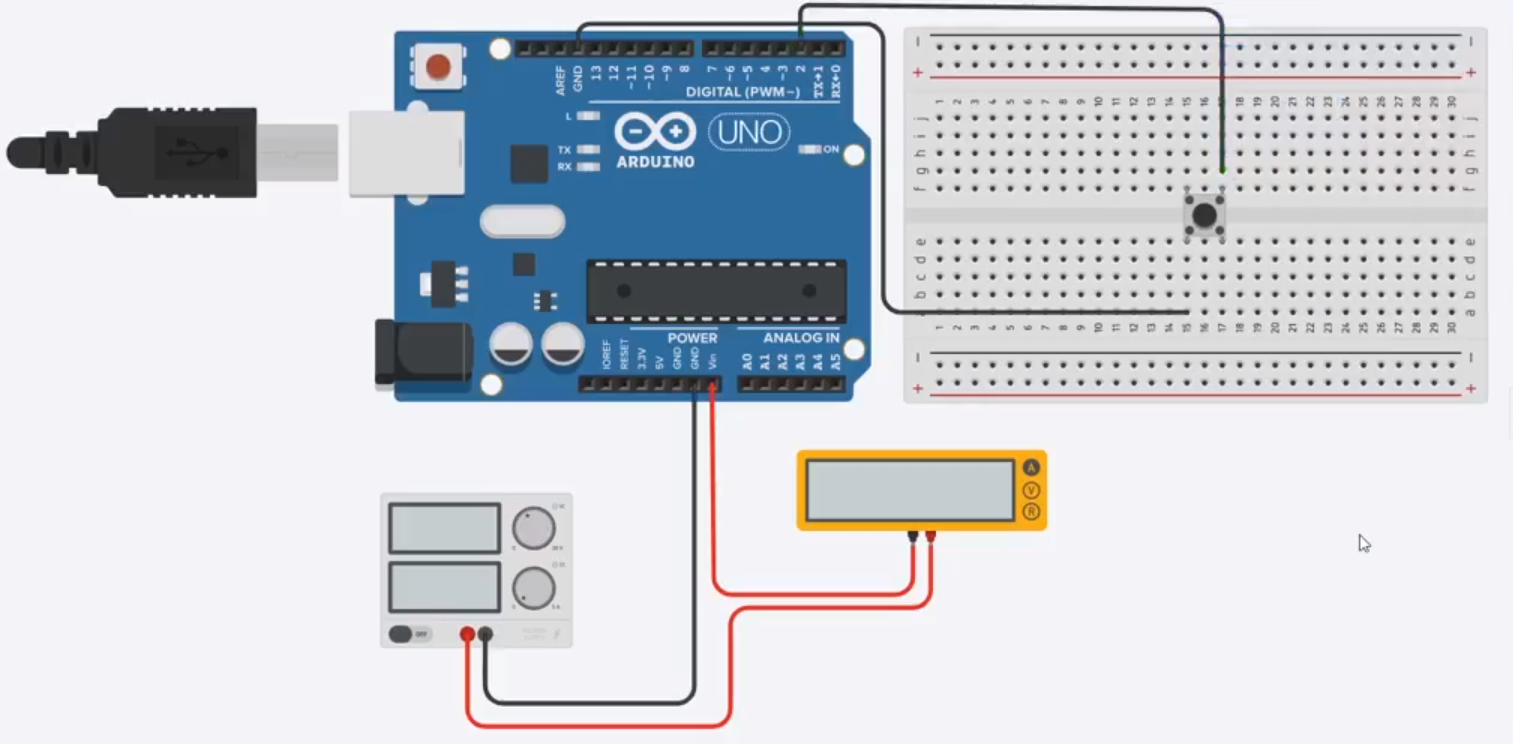
Slika 3.2. Popratno sklopovlje „Deep sleep“ modova [2]

U prethodnoj slici križićima je za svaki mod označeno sklopovlje koje određeni mod koristi. Također iz slike je vidljivo da „Power Down“ mod koristi najmanje sklopovlja te upravo iz tog razloga ovaj mod omogućava najveću uštedu energije. Svakako pažnju treba obratiti na to koje je sklopovlje za određeni mod isključeno. Npr. ako naš projekt sadrži potrebu za analogno digitalnom pretvorbom, sve će biti uredu ako koristimo „Idle“ mode, dok neće biti uredu ukoliko koristimo „Power down“ mod jer, vidljivo iz prethodne slike „Power down“ mod ne podržava ADC odnosno analogno digitalnu pretvorbu.

Svakako kada mikrokontroler ode u „spavanje“ on će spavati sve dok ga ne probudimo. Za naš projekt vršit ćemo buđenje pomoću vanjskog podražaja (eng. Interrupt). Vanjski podražaji funkcioniraju na način promjene napona na određenom pinu. Programski učinimo da na „Interrupt“ pinu vlada napon više logičke razine odnosno 5V. Kada na taj pin dovedemo preko tipkala GND, napon na „Interrupt“ pinu pada na nižu logičku razinu odnosno došlo je do promjene napona na datom pinu koju mikrokontroler razumije kao naredbu za buđenje iz mode za spavanje.

## 3.3 ELEKTRIČNA SHEMA SPAJANJA

Način na koji su pojene komponente u projektu prikazan je na sljedećoj slici.



Slika 3.3. Električna shema spajanja projekta

# 4.TEST

Prikazi rezultata dani su u sljedećoj tablici.

Tablica 4.1 Rezultati mjerenja potrošnje za određene stadije

|  |  |
| --- | --- |
| **STADIJI OPTIMIZACIJE** | **POTROŠNJA** |
| POTROŠNJA NA POČETKU | 58 mA |
| PRESKALIRANJE S 2 | 54 mA |
| PRESKALIRANJE S 4 | 51 mA |
| PRESKALIRANJE S 8 | 49 mA |
| PRESKALIRANJE S 16 | 48 mA |
| SLEEP\_MODE\_ADC | 45 mA |
| SLEEP\_MODE\_PWR\_SAVE | 38 mA |
| SLEEP\_MODE\_STANDBY | 38 mA |
| SLEEP\_MODE\_PWR\_DOWN | 37 mA |
| UKLANJANJE NAREDBI DigitalWrite() | 32 mA |
| UKLANJANJE LEDICE S MIKROKONTROLERA | 4.6 mA |
| UKLANJANJE SVIH LEDICA | 0.01 mA |

Svim navedenim postupcima uštedili smo energiju 5800 puta, odnosno ako bi naš sustav napajan baterijama inače funkcionirao jedan dan, primjenjujući ove metode uštede potpuno hardverski identičan sustav mogao bi samostalno funkcionirati minimalno 10 godina, odnosno u idealnim uvjetima gotovo 16 godina.

# 5. ZAKLJUČAK

Prilikom uštede energije preskaliranjem frekvencije preskaliranjem većim od 8 dolazi do beznačajno male uštede u odnosu na performanse mikrokontrolera tako da je u realnim uvjetima bolje ostaviti mikrokontroler da radi znatno brže nego uštedjeti minimalnu količine energije.

Prilikom korištenja „Deep sleep“ modova potrebno je paziti da li odeređeni mod podržava sve funkcionalnosti koje su u našem projektu potrebne. Svakako je potrebno obratit pažnju na način „buđenja“ mikrokontrolera iz „Deep sleep“ modova kako bi sam mikrokontroler mogao izvršiti potrebne funkcije za koje je predviđen.

Načini uštede energije se uglavnom koriste za baterijski napajane sustave kod kojih je potrošnja svakog mili-Ampera značajno, odnosno senzorske ustave koji se ne mogu napajati iz mreže te se relativno udaljeno nalaze od mjesta kojima često obitavamo.

Ovi načini uštede su itekako korisni što potkrepljuje činjenica da sustav koji je mogao samostalno funkcionirati smo jedan dan, nakon optimizacije potrošnje može funkcionirati 10 godina što je vrlo impresivan rezultat.

# 6. PROGRAMSKI KOD

#include <avr/sleep.h>

int wakePin = 2; // pin used for waking up

void wakeUpNow() {

// execute code here after wake-up before returning to the loop() function

// timers and code using timers (serial.print and more...) will not work here.

// we don't really need to execute any special functions here, since we

// just want the thing to wake up

}

void sleepNow() {

//Potrošnja na početku 58 mA

// CLKPR=0x01; Potrošnja 54 mA

// CLKPR=0x02; Potrošnja 51 mA

// CLKPR=0x03; Potrošnja 49 mA

// CLKPR=0x04; Potrošnja 48 mA

// SLEEP\_MODE\_ADC Potrošnja 45 mA

// SLEEP\_MODE\_PWR\_SAVE Potrošnja 38 mA

// SLEEP\_MODE\_STANDBY Potrošnja 38 mA

// SLEEP\_MODE\_PWR\_DOWN Potrošnja 37 mA

// Do sada ušteđeno energije 63.8%

//Uklanjanjem naredbi digitalWrite() uštadjeli smo dodatnih 5 mA odnosno potrošnja iznosi 32 mA

//Uklanjanjem ledice a Arduino-a potrošnja je pala na svega 4.6 mA

//Uklanjanjem i druge ledice potrošnja bi pala na otprilike 0.01 mA

//Što iznosi ušteda energije od 5800 puta :)

//SLEEP\_MODE\_IDLE – najmanja ušteda

//SLEEP\_MODE\_ADC

//SLEEP\_MODE\_PWR\_SAVE

//SLEEP\_MODE\_STANDBY

//SLEEP\_MODE\_PWR\_DOWN – najveća ušteda

set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN); // sleep mode is set here

sleep\_enable(); // enables the sleep bit in the mcucr register

attachInterrupt(0,wakeUpNow, LOW); // use interrupt 0 (pin 2) and run function

sleep\_mode(); // here the device is actually put to sleep!!

// THE PROGRAM CONTINUES FROM HERE AFTER WAKING UP

sleep\_disable(); // first thing after waking from sleep: disable sleep...

detachInterrupt(0); // disables interrupt 0 on pin 2 so the wakeUpNow code will not be executed during normal running time.

}

void setup() {

CLKPR=0x80;

CLKPR=0x04;

//pinMode(led,OUTPUT);

pinMode(wakePin, INPUT\_PULLUP);

attachInterrupt(0, wakeUpNow, LOW); // use interrupt 0 (pin 2) and run function wakeUpNow when pin 2 gets LOW

}

void loop() {

sleepNow(); // sleep function called here

}

# 7. REFERENCE

1. Atmel : “ATMega328P Datasheet“ s Interneta, <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf> , 18.01.2020.
2. Ilett, Jullian : “Arduino power saving tutorial: Clock Pre-scale and sleep“ s Interneta, <https://www.youtube.com/watch?v=iMC6eG24S9g> , 18.01.2020.
3. Make Course : “Tutorial: Using Power Saving Sleep Modes on the Arduino“ s Interneta, <https://www.youtube.com/watch?v=HiAbxSO_9nU> , 18.01.2020.