SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br.4237

NEINVAZIVNI SUSTAV ZA MJERENJE KVALITETE SNA

Martin Sertić

Zagreb,lipanj 2015.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 13. ožujka 2015.

ZAVRŠNI ZADATAK br. 4237

Pristupnik: **Martin Sertić (0036459089)**
Studij: Računarstvo
Modul: Računalno inženjerstvo

Zadatak: **Neinvazivni sustav za mjerenje kvalitete sna**

Opis zadatka:

U okviru završnog rada potrebno je razviti neinvazivni sustav za automatsko praćenje kvalitete sna tijekom jedne noći korištenjem ugrađenog računala iz porodice MPS430 Texas Instruments. Odabrati pogodnu razvojnu platformu poput eZ430-Chronos s baterijskim napajanjem i odgovarajućim senzorima pomoću kojih je moguće pratiti aktivnost osobe tijekom noći. Sustav mora bilježiti relevantne događaje uz pohranu vremenske oznake te prikupljene i pohranjene događaje poslati nadređenom PC računalu na analizu. Algoritam prilagoditi tim učinkovitijoj pohrani podataka u memoriju sustava ograničenog kapaciteta i tim manjem utrošku energije za bežično slanje podataka. Razviti jednostavnu aplikaciju za nadređeno računalo za prikaz snimljenih događaja i ocjenu kvalitete sna.

Zadatak uručen pristupniku: 13. ožujka 2015.
Rok za predaju rada: 12. lipnja 2015.

Mentor:
Prof. dr. sc. Davor Petrinović

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Marin Golub

Dekan:
Prof. dr. sc. Damir Sersić

Sadržaj

1. Uvod

Spavanje je sastavni dio svačijeg života. Spavamo svaki dan ili bi barem trebali. Osoba koja je dugo vremena budna postane razdražljiva, slaba, a nije rijetko da dolazi I do glavobolje. Vjerovanje da spavamo kako bi se naše tijelo odmorilo je zastarjelo jer san gotovo da I ne štedi energiju, ali je dokazano da su procesi koje naše tijelo obavlja u snu jednako bitni kao I oni koje obavlja za vrijeme dana. Zato treba spavati svaki dan. Optimalno vrijeme spavanja je 7-8 sati dnevno. Sve što je manje od 6 sati ili veće od 10, smatra se nezdravim. S godinama prosječno vrijeme sna pada, ljudi manje spavaju, ali uvijek bi trebali naći dovoljno vremena za san jer dobar I kvalitetan san poboljšava kondiciju I može produžiti život. Od 15 do 35 posto ljudi pati od loše kvalitete sna. Kvaliteta sna ovisi o više stvari. Postoje neki objektivni pokazatelji kao što su latencija uspavljivanja, broj buđenja i trajanje budnosti, a s druge strane imamo I subjektivne pokazatelje kao na primjer dubina spavanja I osjećaj odmorenosti nakon spavanja.

eZ430-Chronos je jedna vrsta ugradbenog računarnog sustava. U današnje doba mi smo okruženi ugradbenim sustavima, vidimo ih na svakom koraku, od digitalnih satova, mp3 playera pa sve do mobitela. eZ430-Chronos je sat koji na sebi ima razne senzore za mjerenje akceleracije, tlaka, temperature. Budući da ove objektne pokazatelje kvalitete sna možemo mjeriti, u tu svrhu nadogradit ćemo sustav eZ430-Chronos.

2. Struktura sustava

Sustav za mjerenje kvalitete sna ostvaren je koristeći Texas Instruments eZ430-Chronos.

2.1. Značajke sustava

eZ430-Chronos je visoko integrirano I bežično razvojno okružje bazirano na CC430F6137 mikrokontroleru.

Slika 1. eZ430-Chronos sat s USB pristupnom točkom I USB programerom

eZ430-Chronos je kompletni razvojni sustav koji se sastoji od LCD zaslona sa 96 segmenata, integriranog senzora pritiska, troosnog akcelerometra za kontrolu pokreta. Integrirano bežično sučelje omogućuje eZ430-Chronosu da djeluje kao centralna stanica za sve okolne bežične senzore kao što su pedometri I monitori otkucaja srca.

eZ430-Chronos može se raskopati kako bi bio reprogramiran nekim vlastitim aplikacijama za što se dobije USB programmer.

Radi se o ultra-low power sustavima koji troše vrlo malo električne energije i time omogućavaju što dulje trajanje baterije. Ovisno o načinu upotrebe uređaja, životni vijek baterije se može značajno skratiti.

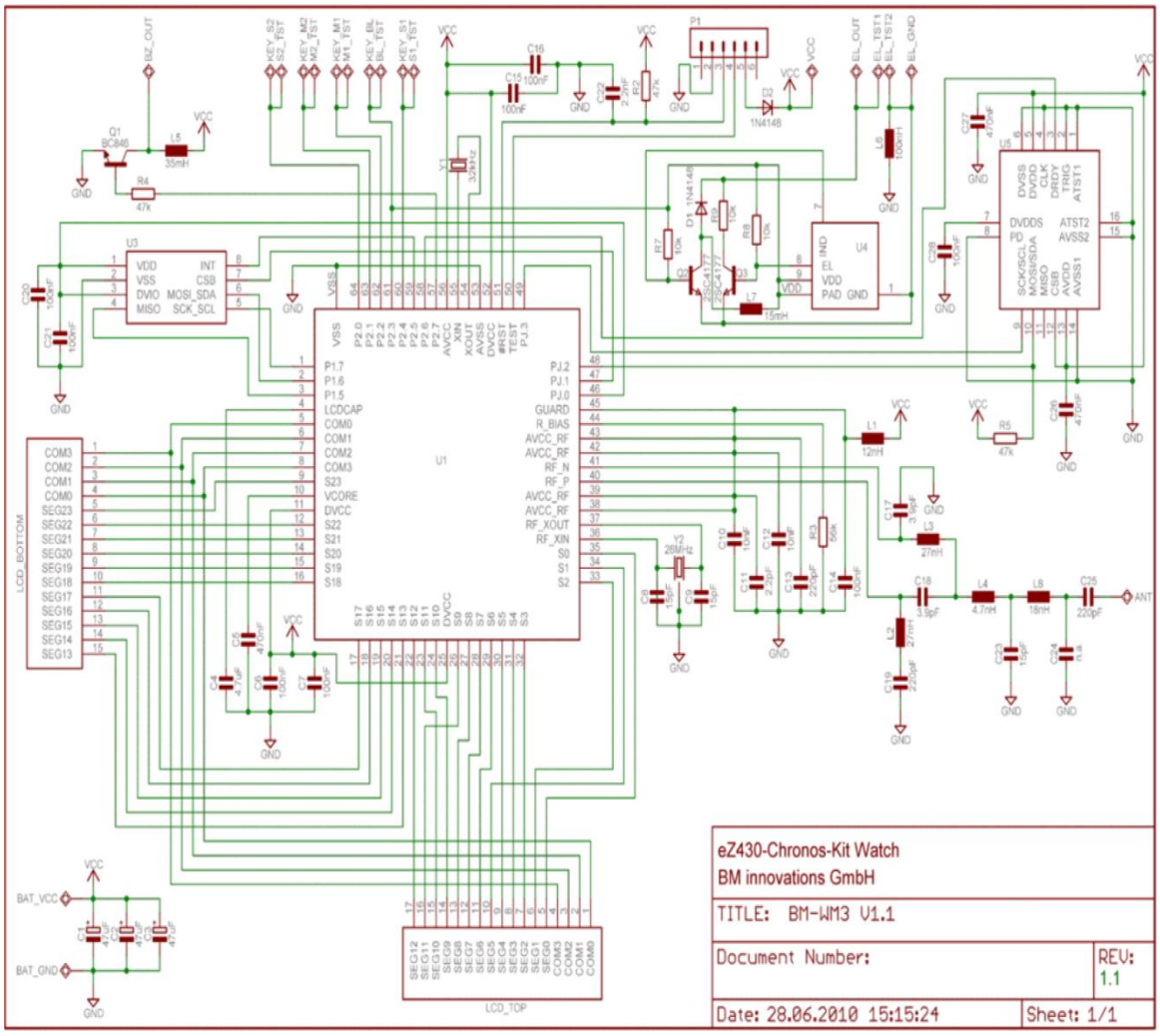
Tablica .1 Potrošnja električne energije I trajanje baterije u ovisnosti o načinu rada

Mode	Average Current	Battery Life
Shelf mode (LPM4)	2.7 µA	92.6 months
Welcome screen (LPM3)	8.9 µA	28.0 months
Time and date	9.0 µA	27.7 months
Continuous temperature measurement	10.0 µA	25.0 months
Continuous altitude measurement	18.0 µA	13.8 months
Continuous acceleration measurement	166.0 µA	1.5 months
Continuous BlueRobin RX	40.0 µA	6.2 months
Continuous SimpliciTI PPT (no button pressed)	10.0 µA	25.0 months
Continuous SimpliciTI SYNC	0.9 mA	8 days
Continuous SimpliciTI ACC	3.7 mA	2 days
1 hour per day BlueRobin RX	10.3 µA	24.2 months
1 hour per day SimpliciTI PPT (no button pressed)	9.1 µA	25.4 months
1 hour per day SimpliciTI SYNC	46.1 µA	5.4 months
1 hour per day SimpliciTI ACC	169.9 µA	1.4 months

3. eZ430-Chronos Hardware

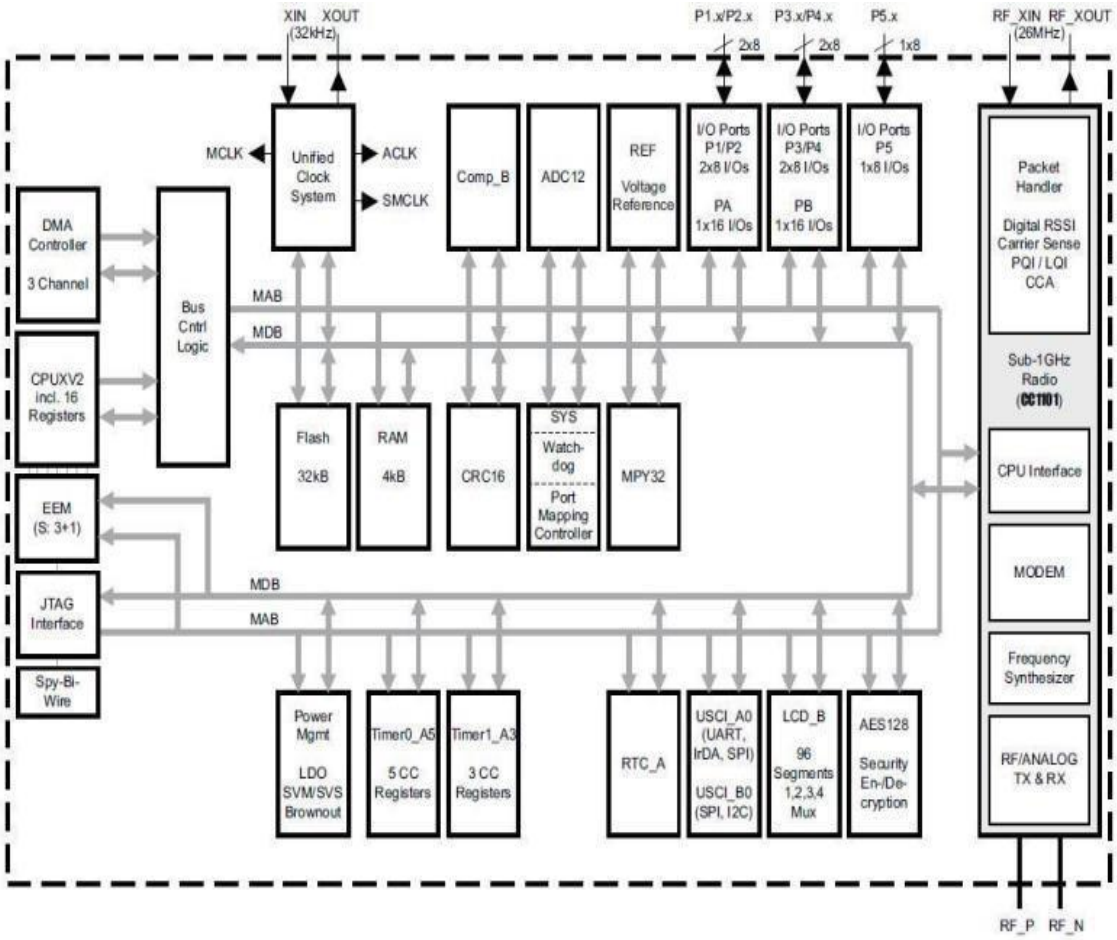
U ovom poglavlju bit će opisan hardware eZ430-Chronos sata, koji je glavni dio razvojne platforme.

3.1. eZ430-Chronos sat



Slika 2. Shema eZ430-Chronos sata

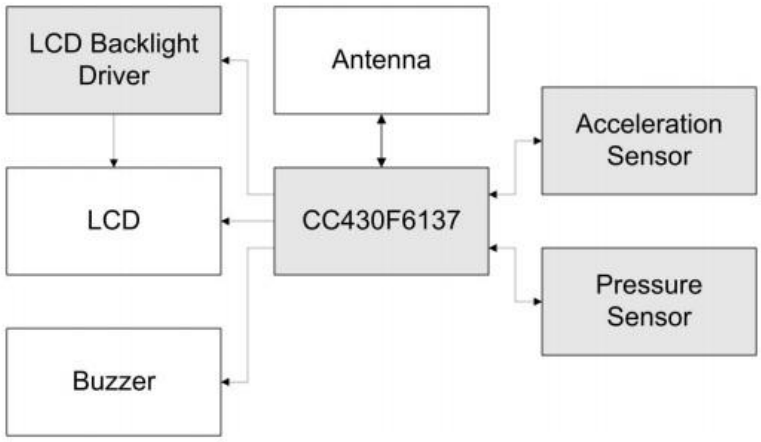
Cijeli sustav eZ430-Chronos se temelji na CC430F6137 mikrokontroleru iz porodice CC430 procesora smanjene potrošnje energije (ultra-low-power microcontroller) i integriranoj radiofrekvencijskoj primopredajnoj jezgri na jednoj SoC (System on Chip) tiskanoj pločici. Na slici 3 možemo vidjeti strukturu mikrokontrolera CC430F6137.



Slika 3 Struktura mikrokontrolera CC430F6137

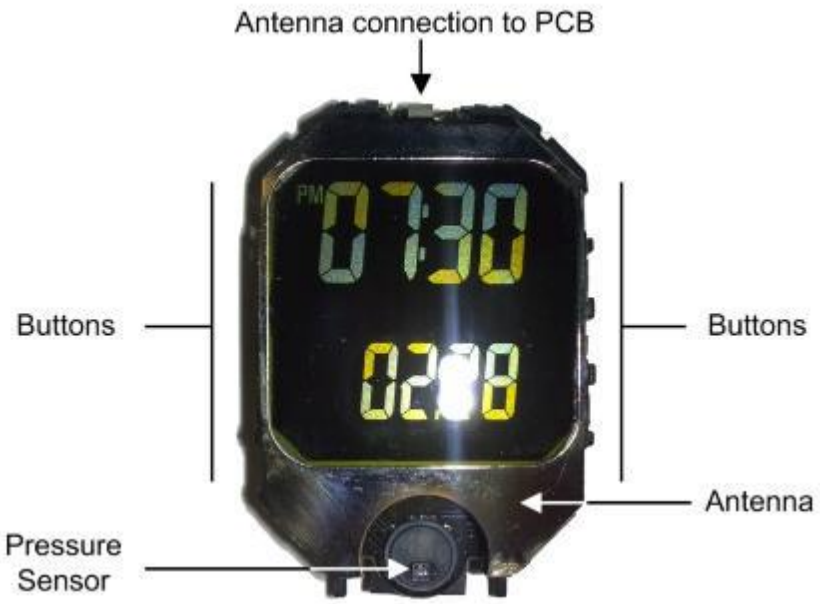
Porodica CC430 osigurava visok stupanj integracije između jezgre mikrokontrolera, softwarea, perifernih djelova I RF odašiljača omogučujući tako tim SoC (System on Chip) rješenjima laku primjenu.

Porodica CC430F61XX su SoC mikrokontroleri koji kombiniraju odličnu performansu izvanrednog <1GHz RF odašiljača skupa sa MSP430 CPUXV2, do 32KB programirljive flash memorije, do 4KB RAM-a, dva 16-bitna timera, 12-bitnim AD (analog to digital) converterom, unutarnjim temperaturnim I baterijskim senzorima, komparatorom , 128-bitni AES sigurnosnim acceleratorom,podrškom za USB, RTC (Real Time Clock) s mogučnostima alarma, LCD driverom I imaju do 44 ulazno izlaznih priključaka.



Slika 4. Blok dijagram eZ430-Chronos sata

Slika 5 pokazuje izgled Chronos sata izvan kučišta. PCB ne koristi antenu, nego metalni okvir na satu koji okružuje LCD ima ulogu antene. Antena ima najbolje performance kada je sat na ruci. Prijenos podataka obavlja se na frekvencijama nižim od 1GHz, a eZ430-Chronos dolazi u nekoliko varijanti.



Slika 5. Vanjsko sklopovlje na eZ430-Chronos satu

Uređaj u sebi ima 96-segmentni LCD display koji se pokreće od strane mikrokontrolera. Svaki segment može biti individualno kontroliran od strane software. Display ima osvjetljenje koje je također potpuno kontrolirano od strane software.

Uređaj je opremljen s VTI CMA3000-D01 troosnim MEMS akcelerometrom, koji ima doseg do 8g I frekvenciju uzorkovanja od 400Hz. Komunicira s mikrokontrolerom koristeći SPI sabirnicu (Serial Peripheral Bus).

Također ima VTI SCP1000-D11 senzor pritiska koji ima domet od 30kPa do 120kPa i može biti upotrebljen za određivanje visine iznad razine mora. Komunicira s mikrokontrolerom pomoću I2C sabirnice (Inter-Integrated Circuit).

Uređaj također ima zujalicu (buzzer) koji je koristan kad treba signalizirati pritisak neke tipke ili kad želimo simulirati alarm. Njime je također moguće upravljanje putem software.

Kao što vidimo u tablici 2, sat dolazi u 3 različite varijante. Radi se o frekvenciji od 433 MHz koja je sloboda za rad u gotovo svim državama svijeta, frekvenciji od 868 MHz u Europi I Indiji I frekvenciji od 915 MHz koja je za tržište Sjeverne I Južne Amerike. U ovom završnom radu korištena je inačica sata eZ430-Chronos-868 gdje se sav radiofrekvencijski prijenos obavljao na frekvenciji od 868 MHz.

Tablica 2. Različite varijante eZ430-Chronosa s obzirom na radnu frekvenciju RF odašiljača

	RF frekvencija rada	Regija
eZ430-Chronos-433	433 MHz	Cijeli svijet
eZ430-Chronos-868	868 MHz	Europa I Indija
eZ430-Chronos-915	915 MHz	Sjeverna I Južna Amerika

3.2. eZ430-Chronos pristupna točka

Pristupna točka CC1111 od Texas Instruments je prvi SoC (System on Chip) primopredajnik s integriranim USB kontrolerom koji omogućuje lagani prijenos podataka između RF pristupne točke I PC-a. Može doći u više varijanti, tako da flash memorija može imati 8, 16 ili 32 KB, a RAM-a može biti 1, 2 ili 4 KB.

Slika 6. CC1111 pristupna točka

Pristupna točka omogučuje bežičnu komunikaciju s eZ430-Chronosom direktno iz PC-a kako bi skinuli podatke s njega, sinkronizirali informacije, postavili sat u određeni mod (vezano za datalogger, bit će govora o tome kanije) ili kako bi upravljali nekim programima koji se izvode na PC-u pomoću sata (npr. Power point prezentacijom).

Za programiranje pristupne točke može se koristiti CC Debugger koji se s 10-pinskim kabelom spaja na CC1111 pločicu preko konektora koji se inače dobije s uređajem (u mojem paketu nije bilo ni CC Debuggera ni konektora). Potrebno je zalemiti priključke CC1111 na CC Debugger da bi se izvelo prevođenje.

Nakon što je zalemljeno, moze se programirati kroz SmartRF Flash Programmer program. Prevođenje programa moguće je samo u IAR Embedded Workbench. Nakon uspješnog programiranja, potrebno je odlemiti sve veze I isprobati program. Na slici 6 možemo vidjeti način spajanja CC1111 pristupne točke na CC Debugger.

CC-Debugger	RF Access Point
Pin 1	GND
Pin 2	Vcc
Pin 3	P2.2 (DC)
Pin 4	P2.1 (DD)
Pin 7	RST
Pin 9	Vcc

Slika 7. Spajanje CC Debuggera na pristupnu točku

3.3. eZ430-Chronos USB programer

Za programiranje sata koristi se eZ430-Chronos sučelje za programiranje koje dolazi u paketu sa satom. USB sučelje spaja se direktno na eZ430-Chronos sat preko 6 priključaka kao što možemo vidjeti na slici 8.

Slika 8. Primjer spajanja Chronos sata na USB programer

Za pisanje programa I prevođenje postoje dva službena razvojna okruženja koja su podržana od strane platforme, a to su CSS (Code Composer Studio) firme Texas Instruments I IAR Embedded Workbench. Svo programiranje za ovaj završni rad rađeno je u Code Composer Studiu (CCSv6.1.0).

Code Composer Studio podržava 2 prevoditelja, a to su Texas Istruments prevoditelj I GCC MSP prevoditelj. Za ovaj završni rad korišten je TI prevoditelj. Prvotno ima ograničenje koda, no nakon što se na Texas Instruments stranici zatraži 3-mjesečna dozvola za evaluaciju to ograničenje se ukida I moguć je normalan rad.

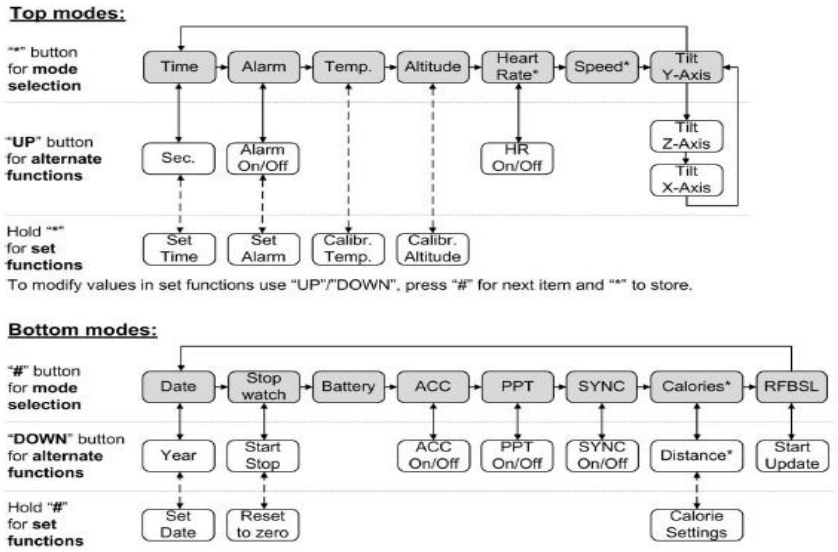
Nakon što se USB programer sa spojenim eZ430-Chronos satom spoji preko USB veze s računalom te željeni program prođe postupak prevođenja, Code Composer Studio preko USB veze prenosi program na sat iz kojeg je nužno izvaditi bateriju prije početka programiranja kako nebi došlo do oštečenja.

4. eZ430-Chronos software

Ovdje će biti opisan software koji je uključen u eZ430-Chronos paket. Postoje dva različita načina rada sata, a to su Chronos Sports watch I Chronos Data logger način rada. Svaki od ta dva načina rada koristi po jednu windows aplikaciju za komunikaciju sa satom. Chronos Sports watch koristi Chronos Control Center PC software dok Chronos Data logger koristi Data logger PC software.

4.1. Sports watch

Sat dolazi programiran u Sports watch načinu rada koji ima široko područje mogućnosti. Osim osnovnih funkcija svih satova, kao što su vrijeme, datum, alarm I štoperica imamo malo naprednije mogućnosti kao mjerač visine, monitor otkucaja srca, sustav koji mjeri potrošnju kalorija. Zaslon sata sastoji se od dvije linije. Za kretanje po opcijama gornje linije koristi se tipka označena s „*“, dok se za kretanje po opcijama donje linije koristi tipka „#“. Svaki od načina rada ima i alternativnu funkciju koja se pokreće s tipkom „UP“ za gornju liniju ili tipkom „DOWN“ za donju liniju. Slika 9 prikazuje načine rada Sports watch programa.

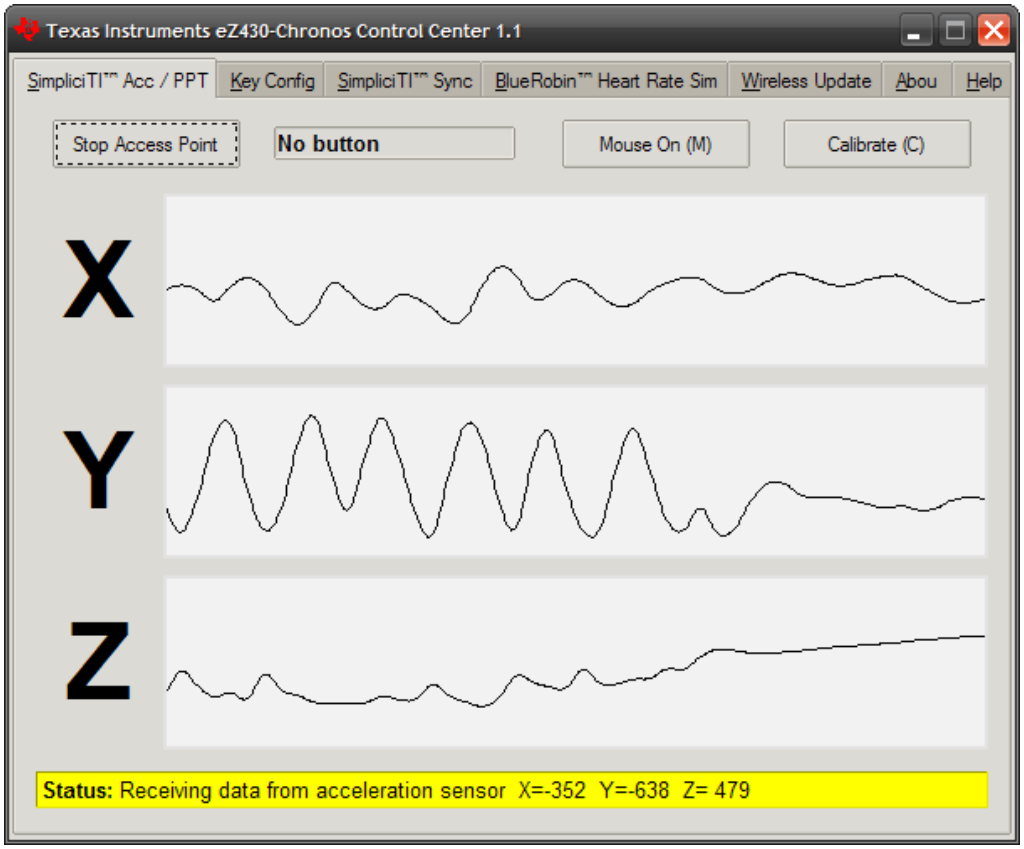


Slika 9. Načini rada Sports watch

Chronos Control Center omogućuje programeru da vidi većinu funkcionalnosti Chronos sata u akciji. Na slici 10 možemo vidjeti izgled Chronos Control Centera.

Chronos Control Center se bežično spaja sa satom I omogućuje sljedeće funkcije:

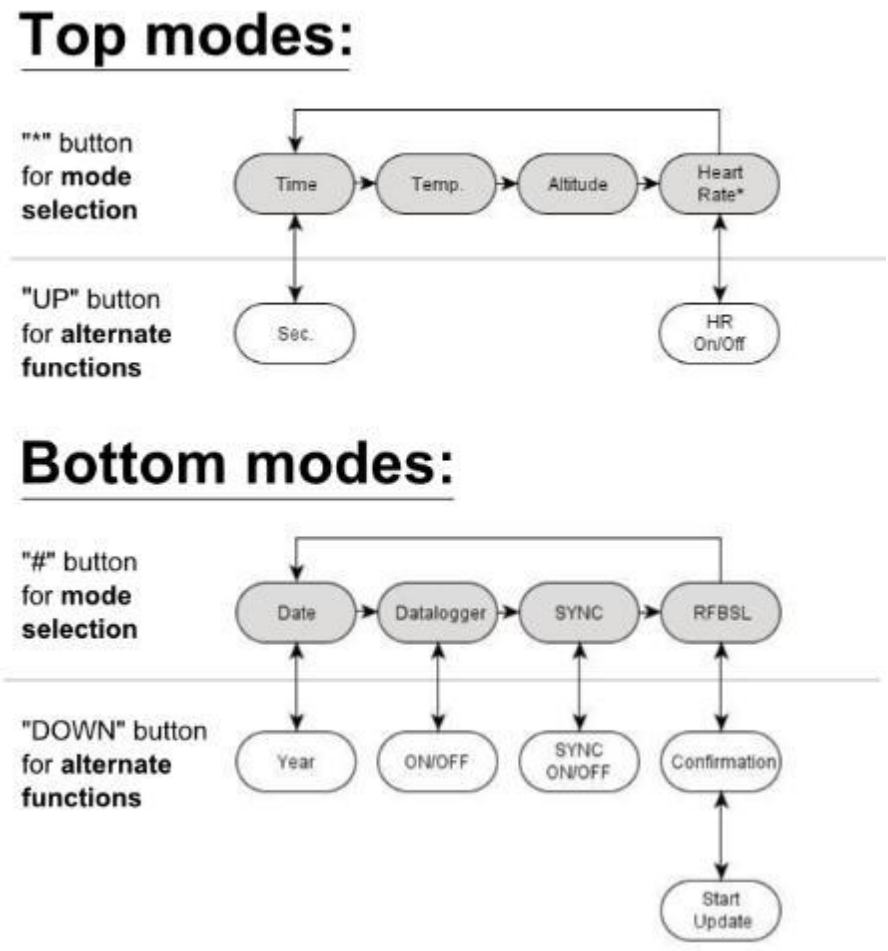
- 3D graf akceleracije
- bežično upravljanje mišem
- bežično upravljanje power point prezentacijom
- korištenje gumbi sa sata kao shortcut za neke tipke tipkovnice
- sinkronizacija vremena I datuma s računalom
- simulator otkucaja srca I simulator brzine
- bežični update firmwarea sata



Slika 10. Chronos Control Center

4.2. Data logger

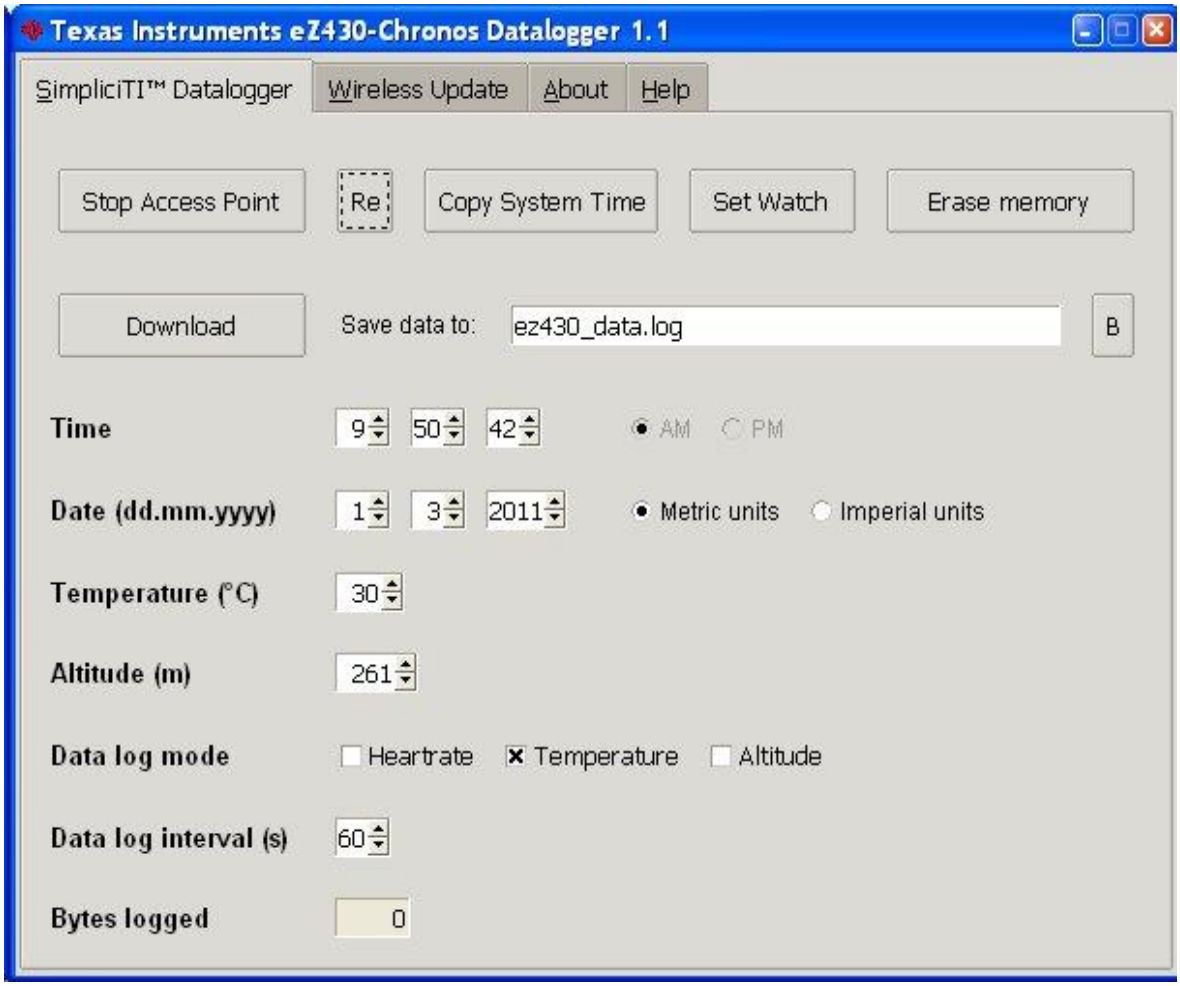
Osim Sports watch, sat dolazi s dodatnim firmwareom koji pretvara sat u data logger. Sat je u mogućnosti logirati temperature, nadmorsku visinu I otkucaje srca u intervalima od 1 do 255 sekundi. 8 KB flash memorije je rezervirano za logiranje podataka od nekoliko sati pa I do par dana, ovisno o interval očitanja odnosno logiranja. Pohranjeni podaci mogu biti prebačeni na računalo I onda korišteni za daljnje analize. Data logger podržava manje funkcija nego Sports watch jer je dio funkcionalnosti izbrisan kako bi se oslobodila memorija za logiranje podataka. Slika 11 prikazuje načine rada Data logger programa.



Slika 11. Načini rada Data logger programa

Data logger PC software omogućuje čitanje logiranih podataka sa sata. Također može biti upotrebljen za postavljanje sata u određeni mod. Funkcionalnost je manja nego kod Chronos Control Centra jer je velik dio funkcija maknut kako bi se osiguralo dosta prazne memorije za logiranje samih podataka. Na slici 12 možemo vidjeti izgled Chronos Datalogger programa. Dostupna funkcionalnost uključuje:

- sredstvo za sinkronizaciju vremena I datuma
- odabir podataka koje želimo logirati
- čitanje logiranih podataka
- bežični update firmwarea



Slika 12. Chronos Datalogger

5. Rad na projektu

Kako bismo ostvarili naš cilj, a to je izvedba sustava za pračenje kvalitete sna, moramo prvo razmisliti koje su to uopće karakteristike dobrog I kvalitetnog, odnosno lošeg sna. Ima više čimbenika,od temperature sobe u kojoj spavamo, broja otkucaja srca, do kvalitete madraca na kojem spavamo,no po mom mišljenju najosnovniji pokazatelji kvalitete sna su vrijeme spavanja I broj buđenja tijekom noći. Pomoću malo modificiranog eZ430-Chronos firmwarea moći ćemo odrediti vrijeme spavanja I broj pomaka kroz cijelu noć, od kojih čemo onda odvojiti one značajne koji simboliziraju stanje budnosti od onih beznačajnih koji su samo promjena položaja u krevetu.

5.1. Program sata

Sat dolazi od prije isprogramiran u Sports watch načinu rada, ali uz to dobijemo I datalogger firmware koji logira nadmorsku visinu I temperature. Potrebno je samo podesiti firmware sata da umjesto visine I temperature logira akceleraciju u x,y I z osi.

Datalogger dolazi isprogramiran u programskom jeziku c. Glavna funkcija je main fukcija. Na slici 13 možemo vidjeti izgled main fukcije.

```
120 int main(void)
121 {
122     // Init MCU
123     init_application();
124 
125     // Assign initial value to global variables
126     init_global_variables();
127 
128     // Main control loop: wait in low power mode until some event needs to be processed
129     while(1)
130     {
131         // When idle go to LPM3
132         idle_loop();
133 
134         // Process wake-up events
135         if (button.all_flags || sys.all_flags) wakeup_event();
136 
137         // Process actions requested by logic modules
138         if (request.all_flags) process_requests();
139 
140         // Before going to LPM3, update display
141         if (display.all_flags) display_update();
142     }
143 }
```

Slika 13. Način rada main programa

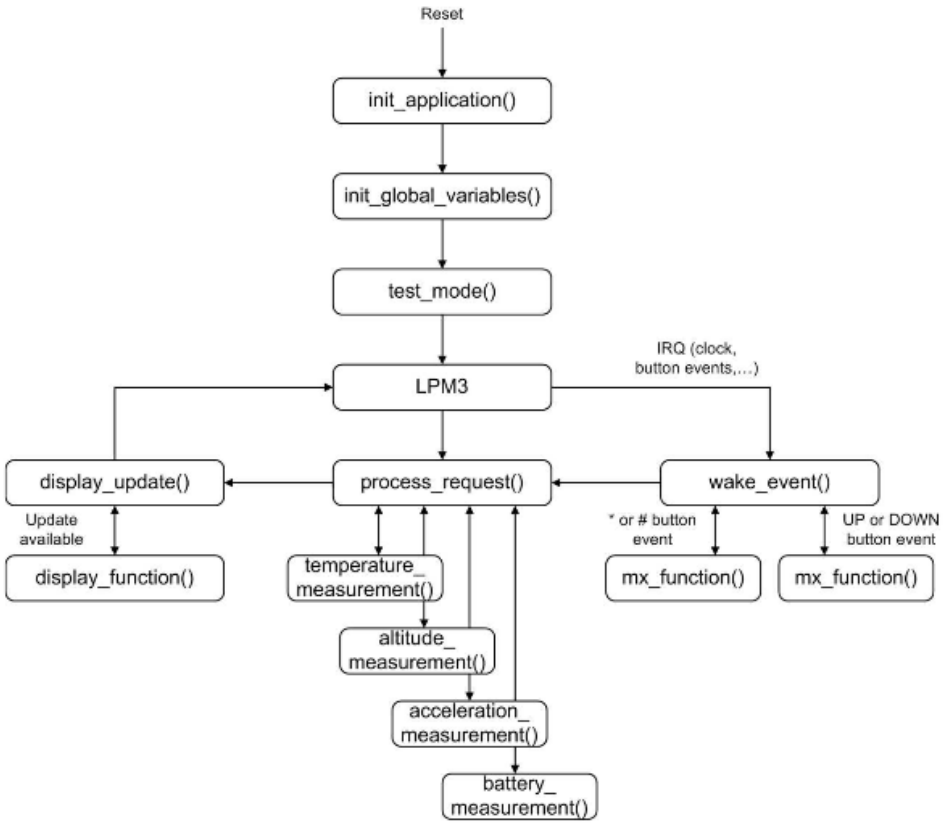
Njenim pokretanjem dolazi do inicijalizacije aplikacije gdje se pokreće watchdog sklop koji resetira funkciju svakih 16s ako dođe do nekog problema. Namješta se frekvencija CPU-a na 12MHz te se zatim resetira radio jezgra I ostale jedinice kao senzor akceleracije, senzor pritiska, LCD zaslon, tipke I timer. Nakon inicijalizacije aplikacije dolazi do inicijalizacije globalnih varijabli gdje se postavljaju pokazivači na gornji I donji menu da prilikom prvog paljenja gornji pokazuje na vrijeme a donji na datalog. Nakon toga se sve zastavice postavljaju u 0 I resetira se LCD zaslon. Također u sklopu inicijalizacije globalnih varijabli dolazi do reseta svih funkcija od vremena I datuma pa sve do mjerača napona baterije.

Nakon završetaka svih inicijalizacija program ulazi u petlju u kojoj ostaje do prekida napajanja sata. Radi u načinu rada smanjene potrošnje sve dok se ne dogodi neki prekid. Ako je doslo do prekida main program izvršava tri linije koda koda koje smo vidjeli na slici 13.

Funkcija wakeup_event ispituje je li I ako je koja je tipka pritisnuta. Ako je tipka „*“ ili „#“ bila kratko pritisnuta, znaci da je zahtijevano kretanje po gornjem, odnosno donjem meniju, a ako je dugo pritisnuta onda se poziva sub menu funkcija iz trenutno aktivne strukture. Ispituje se I zastavica idle_timeout koja govori je li isteklo vrijeme za pritisak tipke.

Nakon što smo obradili sve prekide od tipki ispituje se da li je doslo do prekida jer se zahtijeva neko mjerenje. Funkcija obrađuje 4 različite vrste zahtjeva, a to su mjerenje akceleracije, mjerenje nadmorske visine pomoću senzora za pritisak, pohranjivanje podataka u datalog ili mjerenje napona baterije.

Zadnja fukcija display_update ispituje zastavicu display I tako provjera je li potrebno obnoviti zapis na LCD zaslonu. Nakon što se to obavi, program se vraća u glavnu petlju I u način rada smanjene potrošnje sve dok se ne dogodi neki novi prekid. Na slici 14 vidimo pregledan dijagram toka glavnog main programa.



Slika 14. Dijagram toka main fukcije eZ430-Chronos sata

Osim fukcije main za moj rad je najbitniji bežični protokol simpliciTI. SimpliciTI protokol je bežični protokol za izmjenu podataka I informacija između ez430-Chronos sata I RF pristupne točke. Početak simpliciTI protokola predstavlja funkcija sx_sync koja provjerava da li je dovoljno napajanje baterije I je li trenutno aktivan neki drugi prijenos podataka I ukoliko nije pokreće se simpliciTI u sync načinu rada kao što je vidljivo na slici 15.

```
137 void sx_sync(u8 line)
138 {
139     // Exit if battery voltage is too low for radio operation
140     if (sys.flag.low_battery) return;
141
142     // Exit if BlueRobin stack is active
143     if (is_bluerobin()) return;
144
145     // Start SimpliciTI in sync mode
146     start_simpliciti_sync();
147 }
```

Slika 15. Sx_sync funkcija

Nadalje, funkcija start_simpliciti_sync osvježava zapis LCD zaslona I zatvara dataog ako je bio otvoren. Također da budemo sigurni da smo ušli u sync način rada postavljaju se 3 segmenta LCD zaslona da titraju. Zatim se priprema radio za komunikaciju I postavlja se simpliciti mode u sync u strukturi sRFsmpl.mode.

Poziva se funkcija simplicity_link koja treba inicijalizirati sav potreban hardware za vezu I potom je uspostaviti. Sama funkcija će prvo postaviti zastavicu simpliciti_flag u vrijednost SIMPLICITI_STATUS_LINKING što bi značilo da je započelo uspostavljanje veze. Nakon što se uključi radio jezgra I namjesti snaga na 3.3 dBm postavlja se simpliciti_flag u vrijednost SIMPLICITI_STATUS_LINKED I glavna funkcija simpliciti_link vrača jedinicu.

Nakon uspostave veze pokreće se simplicity_main_sync funkcija. Ona periodički šalje 2B “ready-to-receive” paketa I čeka 0.5s za odgovor. Kada dobije odgovor od RF pristupne točke, ti podaci se dekodiraju I djeluje se s obzirom na odgovor. Ta akcija ovisno o odgovoru pristupne točke I samo dekodiranje odgovora definirana je u funkciji sympliciti_sync_decode_ap_cmd_callback.

Datalog.c je program sata u koji definira ponašanje sata u datalog načinu rada. U header fileu datalog.h definirana su 3 moda dataloger programa, a to su DATALOG_MODE_HEARTRATE, DATALOG_MODE_TEMPERATURE I DATALOG_MODE_ALTITUDE. Za potrebe mjerenja, odnosno logiranja akceleracije dodali smo još jedan mod rada, a to je DATALOG_MODE_ACCELERATION. Ovisno u kojem je modu dataloger sat će bilježiti određenu veličinu. U DATALOG_MODE_ACCELERATION sat bilježi akceleraciju u x, y I z osi. Jedan od bitnijih podataka u datalog strukturi je I interval koji nam govori period očitavanja akceleracije po sve 3 osi, odnosno period popunjavanja datalogera.

Te podatke sat prima od pristupne točke I obrađuje ih u funkciji sympliciti_sync_decode_ap_cmd_callback koja je ranije spomenuta. Ona će nakon što od pritupne točke primi SYNC_UP_CMD_SET_WATCH postaviti parametre sata, a za nas će biti jako bitni mod dataloga I interval. Izgled bitnog dijela funkcije možemo vidjeti na slici 16.

```
case SYNC_AP_CMD_SET_WATCH:     // Set watch parameters

                                    // Data logging mode
                                    sDatalog.mode = simpliciti_data[14];
                                    // Data logging interval
                                    sDatalog.interval = simpliciti_data[15];
```

Slika 16. Postavljanje parametara sata

Automatski čim se postave parametri sata, mod dataloga biti će prebačen u DATALOG_MODE_ACCELERATION za naš slučaj.

Odabirom “DLOG” na donjem meniju sata postavlja se zastavica request.flag.datalog koja signalizira zaslonu da se osvježi I prikazuje slovo R dok god je dataloger aktivan I pokreće se funkcija sx_datalog. Funkcija sx_datalog provjerava je li puna memorija I ako nije poziva funkciju start_datalog koja će uključiti mjerenje akceleracije jer je mod tako postavljen. Na slici 17 možemo vidjeti izgled funkcije start_datalog.

```
void start_datalog(void)
{

    // Start pressure measurement
    if ((sDatalog.mode & (DATALOG_MODE_TEMPERATURE | DATALOG_MODE_ALTITUDE)) != 0)
    {
        // Start altitude measurement
        start_altitude_measurement();
    }

    if ((sDatalog.mode & DATALOG_MODE_ACCELERATION) != 0)
    {
        start_acceleration_measurement();
    }

    // Set datalogger icon
    display_symbol(LCD_ICON_RECORD, SEG_ON_BLINK_OFF);

    // Start data logging
    datalog_sm(NULL, 0, DATALOG_CMD_START);
}
```

Slika 17. Funkcija start_datalog

Funkcija do_datalog ovisno o modu u polje temp sprema podatke, u našem slučaju to su x,y I z akceleracije te to polje temp sprema u buffer kao što možemo vidjeti na slici 18.

```
if (sDatalog.mode == (DATALOG_MODE_ACCELERATION ))
        {
            temp[0] = sAccel.xyz[0];
            temp[1] = sAccel.xyz[1];
            temp[2] = sAccel.xyz[2];
            count = 3;
        }

    datalog_sm((u8*)&temp, count, DATALOG_CMD_ADD_DATA);
```

Slika 18. Dio funkcije do_datalog

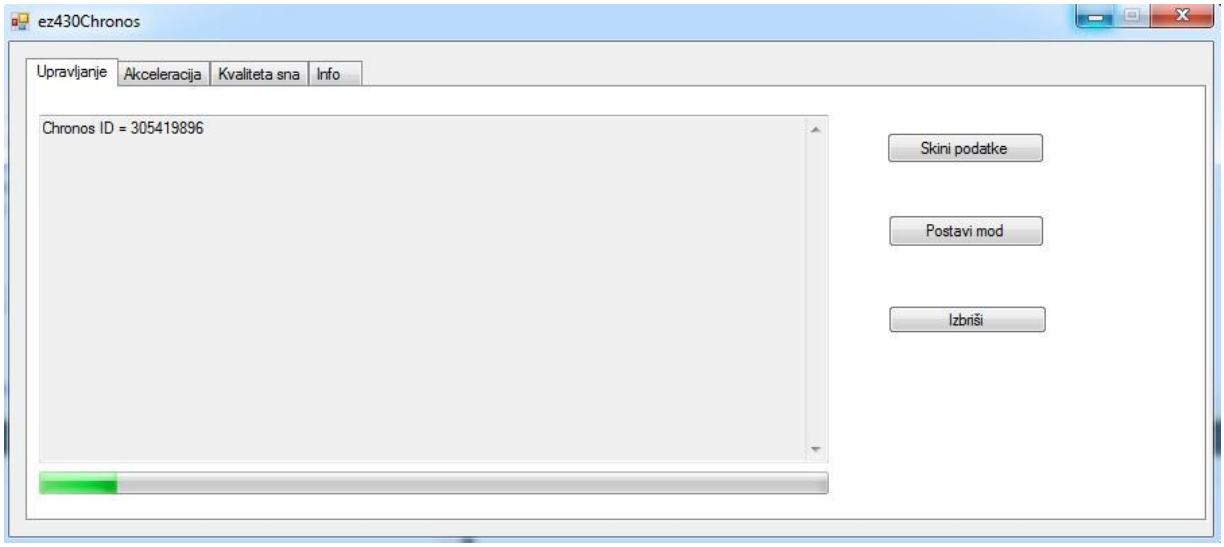
Nakon što smo popunili datalog ili smo pritiskom tipke označili kraj rada, postavlja se zastavica memory_full koja zaustavlja funkciju start_datalog I pokreće stop_datalog. Osvježava se LCD zaslon I zapisuje se buffer u flash memoriju.

Nakon što pristupna točka zatraži prijenos podataka iz flash memorije, oni se prenose I daljnja obrada podataka vrši se u programu pristupne točke.

5.2. Program pristupne točke

Ovo je bio najteži dio projekta, budući da sat dolazi od prije isprogramiran I potrebne su samo male preinake. Program pristupne točke sastoji se od eZ430_Chronos_Net biblioteke koja je dostupna na internet I definira komunikaciju sa satom, ZedGraph biblioteke koja služi za crtanje grafa akceleracije I samog Chronos Downloadera koji je najbitniji. Sve je pisano u programskom jeziku c#.

Chronos downloader prvo izvršava funkciju mainform_Load kojom se otvara glavni prozor aplikacije koji je prikazan na slici 19 te se pomoću eZ430_Chronos_net funkcije openComPort otvara veza za komunikaciju, no pritom sat mora biti u sync načinu rada. Nakon što se otvorila komunikacija, u glavni prozor se ispisuje ID te se pokreće simpliciTI protokol eZ430-Chronos sata.



Slika 19. Izgled glavnog prozora eZ430-Chronos windows programa

U glavnom prozoru imamo 3 tipke I pritisak svake od njih će generirati komunikaciju između programa I sata. Kako bi uopće mogli primiti neke podatke za daljnju obradu, potrebno je postaviti sat u DATALOG_MODE_ACCELERATION jer on inače mjeri I logira samo pritisak, otkucaje srca I temperature.

Pritiskom tipke postavi mod pokreće se funkcija btnSet_Click koja će poslati satu paket od 19B koji će na strain sata obraditi fukcija sympliciti_sync_decode_ap_cmd_callback. Prvi bajt je kontrolni i pomoću njega sat zna da li želimo postaviti parametre, skidati podatke ili brisati memoriju. Dalje ide sati(1B), minute(1B), sekunde(1B), godina(2B), mjesec(1B), dan(1B), 2B prazna, temperature(2B), nadmorska visina(2B), datalog.mode(1B), interval(1B) te su zadnja

3B prazna. Postavljanje parametara sata možemo vidjeti na sljedećoj slici, kao I reakciju funkcije sympliciti_sync_decode_ap_cmd_callback na slici 21.

```
private void btnSet_Click(object sender, EventArgs e)
{
    byte[] data = new byte[19];
    DateTime dateTimeVar = DateTime.Now;
    data[0] = Constants.SYNC_AP_CMD_SET_WATCH;
    data[1] = (byte)(0x80 + (byte)dateTimeVar.Hour);
    data[2] = (byte)dateTimeVar.Minute;
    data[3] = (byte)dateTimeVar.Second;
    data[4] = (byte)((uint)dateTimeVar.Year >> 8);
    data[5] = (byte)((uint)dateTimeVar.Year & 0x00FF);
    data[6] = (byte)dateTimeVar.Month;
    data[7] = (byte)dateTimeVar.Day;
    data[8] = 0;
    data[9] = 0;
    data[10] = 0;
    data[11] = 0xC8;
    data[12] = 1;
    data[13] = 0xF4;
    data[14] = Constants.DATALOG_MODE_ACCELERATION;
    data[15] = 15; //15 seconds interval
    data[16] = 0;
    data[17] = 0;
    data[18] = 0;

    if (_chronos.SendSyncCommand(data))
        textBox1.AppendText("Poslan SET" + System.Environment.NewLine);
    else
        textBox1.AppendText("Nije poslan SET" + System.Environment.NewLine);
}
```

Slika 20. Funkcija btnSet_Click

```
void simpliciti_sync_decode_ap_cmd_callback(void)
{
    u8 i;
    s16 t1, offset;

    // Default behaviour is to send no reply packets
    simpliciti_reply_count = 0;

    switch (simpliciti_data[0])
    {
        case SYNC_AP_CMD_NOP:           break;

        case SYNC_AP_CMD_GET_STATUS:    // Get status
                                        break;
        case SYNC_AP_CMD_SET_WATCH:     // Set watch parameters
                                        break;
        case SYNC_AP_CMD_GET_MEMORY_BLOCKS_MODE_1:
                                        break;
        case SYNC_AP_CMD_GET_MEMORY_BLOCKS_MODE_2:
                                        break;
        case SYNC_AP_CMD_ERASE_MEMORY:  // Erase data logger memory
                                        break;
        case SYNC_AP_CMD_EXIT:          // Exit sync mode
                                        break;
    }

}
```

Slika 21. Funkcija sympliciti_sync_decode_ap_cmd_callback

Nakon što smo satu poslali parametre može se vršiti mjerenje. Prolaskom kroz donji menu može se pokrenuti “DLOG” način rada. Obavljat će se funkcije opisane u prošlom poglavlju I na kraju ćemo dobiti u flash memoriji sata zapise o akceleraciji svakih 15 sekundi jer smo tako podesili interval. U flash memoriju sata bit će upisana vremenska oznaka I stanje x, y I z smjera akceleracije.

Pritiskom tipke skini podatke pokreće se funkcija btnDownload_Click koja će satu pomoću eZ430_Chronos_Net funkcije poslati paket od 19B kao prethodna Set funkcija samo što će ovdje kao prvi bajt biti SYNC_AP_CMD_GET_MEMORY_BLOCKS_MODE_1 dok je tamo bio SYNC_AP_CMD_SET_WATCH. Opet će sa strane sata taj paket obraditi funkcija sympliciti_sync_decode_ap_cmd_callback. Nakon što smo poslali taj paket sat će nam vratiti sadržaj svoje flash memorije. To spremamo u varijablu data koja je polje bajtova I pozivamo funkciju dataparsera da parsira te podatke jer je ovo užasno nepregledan niz bajtova. Dataparser radi na principu konačnog automata stanja. Naime mi znamo u kojem redosljedu

nam podaci stižu od sata I tako prebacujemo dataparser iz stanja u stanje, a svako stanje nam predtsavlja drugu vrstu podataka. Na sljedećoj slici prikazan je redosljed logiranja podataka i koji od njih predstavlja.

```
case DATALOG_CMD_START:
                        if (!sDatalog.flags.flag.on && !sDatalog.flags.flag.memory_full)
                        {// Clear index
                            sDatalog.idx            = 0;
                            sDatalog.delay          = sDatalog.interval;
                            // Add session begin marker to buffer (2 byte)
                            temp = 0xFFFB;
                            datalog_add_to_buffer((u8*)&temp, 2);
                            // Add recording mode to buffer (1 byte)
                            datalog_add_to_buffer((u8*)&sDatalog.mode, 1);
                            // Add recording interval to buffer (1 byte)
                            datalog_add_to_buffer((u8*)&sDatalog.interval, 1);
                            // Add date to buffer (DD.MM.YYYY) (4 bytes)
                            datalog_add_to_buffer((u8*)&sDate.day, 1);
                            datalog_add_to_buffer((u8*)&sDate.month, 1);
                            datalog_add_to_buffer((u8*)&sDate.year, 2);
                            // Add system time to buffer (HH.MM.SS) (3 bytes)
                            datalog_add_to_buffer((u8*)&sTime.hour, 3);
                        }
                        break;

case DATALOG_CMD_CLOSE:   if (sDatalog.flags.flag.on && !sDatalog.flags.flag.memory_full)
                        {
                            // If index is odd, add a dummy byte before writing session end
                            if ((sDatalog.idx & 0x01) == 0x01)  {
                                temp = 0x00;
                                datalog_add_to_buffer((u8*)&temp, 1);}
                            // Add session end marker to buffer (2 byte)
                            temp = 0xFFFE;
                            datalog_add_to_buffer((u8*)&temp, 2);
                            // Write buffer to flash
                            datalog_write_buffer();
                        }
```

Slika 22. Redosljed popunjavanja dataloga

Kako se popunjava buffer takav ćemo redosljed primate od sata. Dakle prva 3 bajta su nebitna, sljedeća 2 su marker za početak, a dalje imamo redom mod(1B), interval(1B), dan(1B), mjesec(1B), godinu(2B), sat(1B), minutu(1B), sekundu(1B), akceleraciju(3B) I oznaku kraja(2B).

Podaci se parsiraju I spremaju u polja. Svaka komponenta akceleracije ima svoje polje, tako da na primjer polje _dataparser.dataAccX [i] predstavlja akceleraciju x osi. Te podaci o

akceleracijama bi trebali u sebi sadržavati statičku akceleraciju sile teže plus još diamičku koja bi došla do izražaja kod gibanja, ali nisam nikako uspio to odvojiti. Kada je sat u mirovanju trebao bi mu modul uvijek biti isti, ali nije. Umjesto toga, koristio sam modul kao sredstvo razlikovanja pomaka od mirovanja.

Logika mog algoritma za određivanje broja buđenja po noći je sljedeća. Izračunati modul svakog zapisa formuli (1).

$$M = \sqrt{accx^2 + accy^2 + accz^2} \qquad (1)$$

I onda ako je (M [z+1] – M [z]) > 4 postavi pomočno polje sa indexom [z +1] u jedinicu. Pomočno polje biti će puno nula I jedinica. Svaka jedinica signalizira da je došlo do nekog značajnijeg pomaka. Ako stoji sama jedinica okružena nulama, to je nebitno. Znači da je netko samo promijenio položaj spavanja. Gledamo samo one slučajeve gdje je gibanje trajalo barem minutu. U tim slučajevima mi imamo 4 pomaka unutar minute, što nam sugerira da je osoba budna. Ako se na primjer osoba tijekom noći digne na wc, što bi automatski značilo da je budna, pomočna varijabla će se puniti jedinicama jer će se osoba kretati. Ako imamo više jedinica za redom znači da se osoba kreće I povečavamo brojač buđenja za jedan. S obzirom a te nule I jedinice možemo odrediti ukupno trajanje sna, kao I ukupan dio vremena koji se proveo budan. Na slikama 23 I 24 vidimo dio algoritma koji računa akceleracije jer su prvotno upisane kao dvojni komplement u satu. Dalje se popunjava polje modula I ovisno o njemu pomočno polje.

```
for (z = 0; z < _dataParser.GetDataCount() ; z++)
{
    if (_dataParser.dataAccX[z] > 128)
        _dataParser.dataAccX[z] = _dataParser.dataAccX[z] - 256;

    if (_dataParser.dataAccY[z] > 128)
        _dataParser.dataAccY[z] = _dataParser.dataAccY[z] - 256;

    if (_dataParser.dataAccZ[z] > 128)
        _dataParser.dataAccZ[z] = _dataParser.dataAccZ[z] - 256;

    m0[z] = Math.Sqrt(Math.Pow(_dataParser.dataAccX[z], 2)
          + Math.Pow(_dataParser.dataAccY[z], 2)
          + Math.Pow(_dataParser.dataAccZ[z], 2));
}
```

Slika 23. Popunjavanje polja modula I računanje ekceleracije

```
pomoc[0] = 0;
for (z = 0; z < _dataParser.GetDataCount()-1; z++)
{
    if (Math.Abs(m0[z + 1] - m0[z]) > 4)
        pomoc[z+1] = 1;
    else pomoc[z+1] = 0;

}
```

Slika 24. Označavanje kretanja u pomočno polje

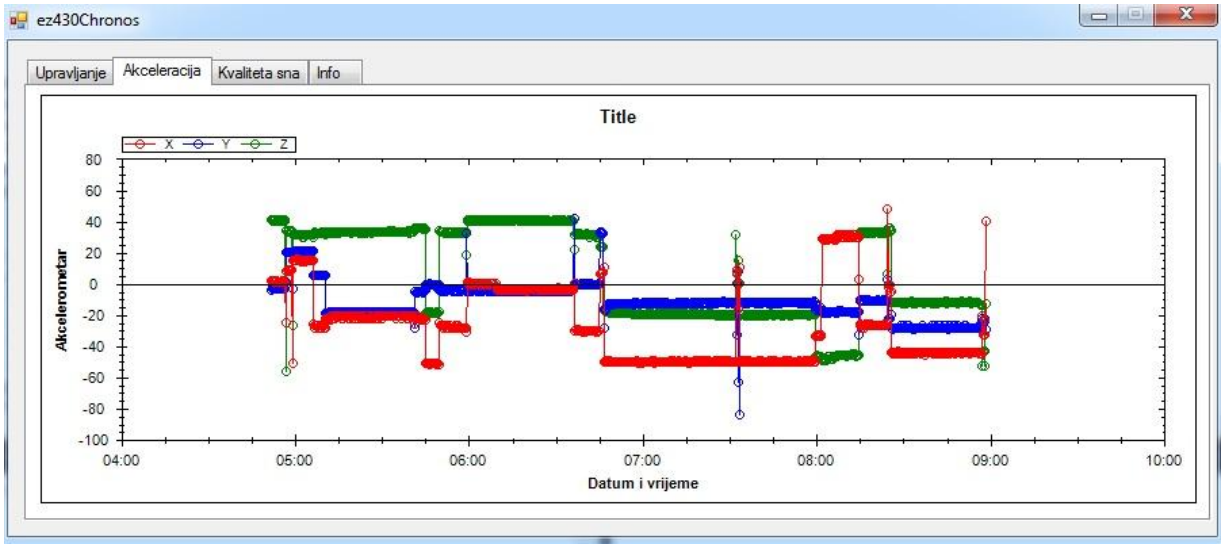
Sljedeča slika prikazuje možda I najbitniji dio algoritma, a to je računanje budnih razdoblja I za svako to razdoblje koje je od minimalno minute povečavamo brojač za jedan.

```
int bzvz=0;
for (z = 0; z < _dataParser.GetDataCount(); z++)
{
    if ((pomoc[z] == 1) && (pomoc[z + 1] == 1) && (pomoc[z + 2] == 1) && (pomoc[z + 3] == 1) &&  (pomoc[z + 4] == 0))
        counter++;
    bzvz = z;
}
```

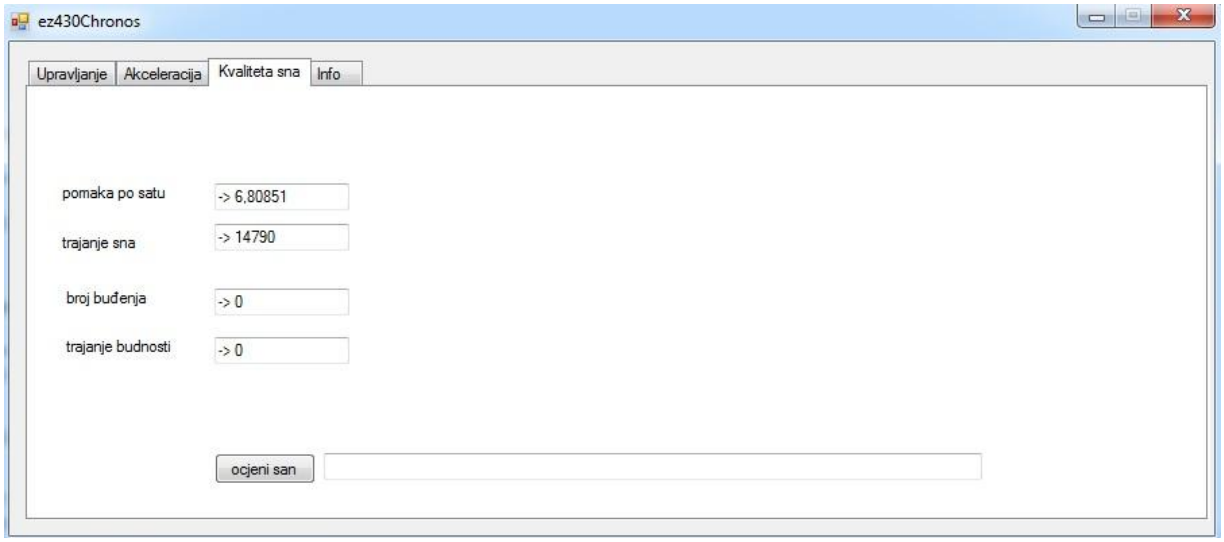
Slika 25. Računanje broja buđenja

Budući da je moguće da je osoba budna, ali da se ne kreće algoritam će svako mirovanje u iznosu od minimalno dvije I pol minute protumačiti kao da je osoba opet zaspala. Ako se osoba pomakne nakon tog vremena, brojač će se opet povećati za 1 makar osoba može I dalje biti budna bez da je uopće zaspala. U dobrom dijelu mojih algoritam se pokazao jako točan kada sam se normalno ponašao, bez da sam išao namjerno probati ga zeznuti.

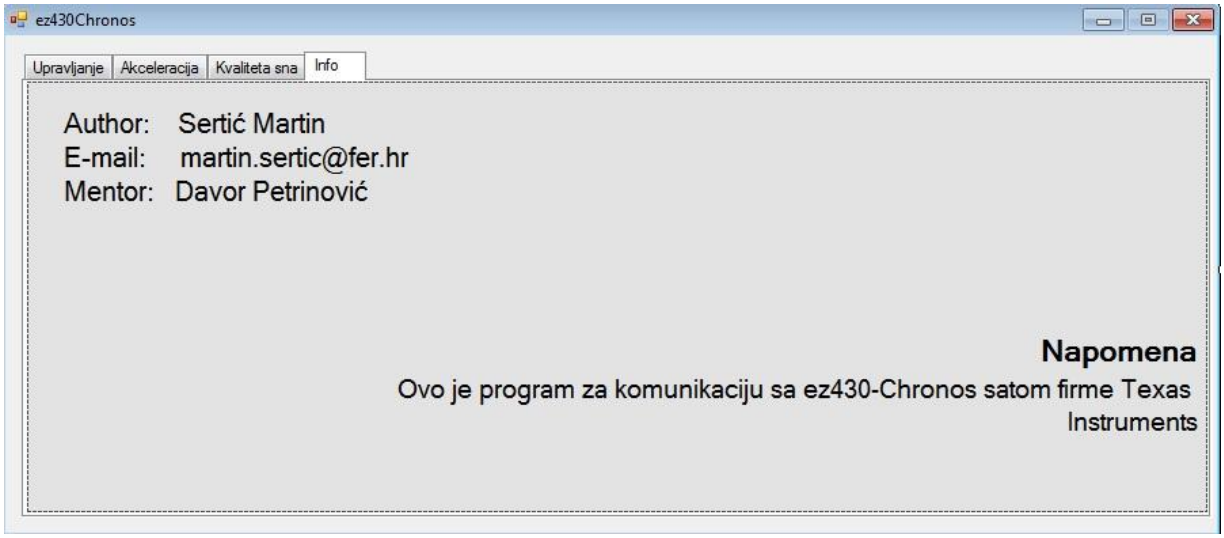
Osim glavnog prozora imamo prozor za grafove akceleracije, prozor koji ocjenjuje kvalitetu sna i imamo još prozor s informacijama o programu. Primjeri tih prozora dani su na sljedeće tri slike .



Slika 26. Prozor s grafom akceleracije



Slika 27. Izgled prozora koji ocjenjuje kvalitetu sna



Slika 28. Izgled Info prozora

6. Zaključak

eZ430-Chronos sustav je korišten za određivanje kvalitete sna. Kvaliteta sna određena je na temelju objektih karakteristika koje se mogu mjeriti senzorima sata kao što su broj buđenja I ukupno trajanje budnosti.

Veliki problem u rješenju je što sat ima ograničenu flash memoriju I može spremati nešto manje od 8kB podataka. Uz svaki podatak o akceleraciji koji je nama potreban, sprema se vremenska oznaka. Zbog nedostatka memorije mogu se logirati podaci otprilike 25 minuta ako se uzme da je interval očitanja jedna sekunda, što bi ukupno dalo oko 1500 podataka koji je svaki veličine 19B. Dalo bi se napraviti sve to puno efikasnije kada bi sat isprogramirali da bilježi podatke u datalog samo ako dolazi do nekog pomaka, no zbog nedostatka vremena I moje osobne neorganiziranosti ja to nisam uspio napraviti.

Također problem sata je visoka potrošnja baterije u sync načinu rada kad se komunicira sa satom odnosno kad sat prima ili šalje podatke. To je dobro napravljeno jer će se tek kad se datalog popuni pokrenuti sync I doći će do prijenosa podataka.

7. Literatura

[1] eZ430-Chronos Development Tool User's Guide (Rev.F)
http://www.ti.com/lit/ug/slau292f/slau292f.pdf

[2] Texas Instruments, Sub-1 GHz RF System-on-Chip with Integrated USB
http://www.ti.com/corp/docs/landing/cc1111/

[3] Texas Instruments Wiki, eZ430-Chronos
http://www.processors.wiki.ti.com/index.php/EZ430-Chronos

[4] Texas Instruments, Debugger and Programmer for RF System-on-Chips
http://www.ti.com/tool/cc-debugger

[5] TI eZ430-Chronos Development Watch- SparkFun Electronics
https://www.sparkfun.com/products/retired/10019

[6] TI Chronos. NET dll, C# example
http://sourceforge.net/projects/chronosexample/files/eZ430%20Chronos%20example/

[7] ZedGraph download| SourceForge
http://sourceforge.net/projects/zedgraph/

8. Sažetak

eZ430-Chronos je vrsta ugradbenog računalnog sustava. U sebi ima mikrokontroler CC430F6137 koji je iz porodice CC430 procesora smanjene potrošnje. Sat radi u načinu rada smanjene potrošnje sve dok se ne dogodi neki događaj kao što je pritisak neke tipke.

Chronos komunicira s pristupnom točkom putem protokola simpliciTI. Postavljanje sata vrši pristupna točka tako da mu pošalje parametre, dok sat šalje logirane podatke pristupnoj točki na daljnju obradu.

Obrada podataka automatski se događa čim oni dođu sa sata I program kao izlaz daje ocjenu kvalitete sna.

9. Summary

eZ430-Chronos is embedded microprocessor system. It has CC430F6137 microcontroller which is part od CC430 family of low power procesors. Watch works in low power mode until it wakes up after a button has been pressed.

eZ430-Chronos communicates with access point using simpliciTI protocol. Seting of the watch does access point program by sending watch parameters, while watch itself is sending logged data to acces point for further analysis.

Data processing starts automatically when data is transfered and as the end result we get sleep evaluation.