Digitalni sat s automatskim podešavanjem vremena na arduino platformi

Begić, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij računarstvo

DIGITALNI SAT S AUTOMATSKIM PODEŠAVANJEM VREMENA NA ARDUINO PLATFORMI

Završni rad

Filip Begić

Osijek, 2017.

SADRŽAJ

1. UVOD

U završnom radu projektirana je ura, sat. Sat mora imati mogućnost automatskog postavljanja vremena. Automatsko postavljanje smanjuje moguće oscilacije vremena s obzirom na ručno podešavanje.

Vrijeme je prikazano na matričnim pokazivačima (8x8 LED matrice) na jednoznačan način gdje se mogu razlikovati sati, minute i sekunde.

Završni rad sadrži Arduino Nano ploču na koju su povezane sve ostale komponente poput GPS modula s pripadajućom antenom, LED (eng. *Light Emitting Diode*) matrica, RTC (eng. *Real Time Clock*) i sl. Sklop može električnu energiju primiti putem USB-a ili adapterom.

Ostatak rada organiziran je na sljedeći način. U drugom poglavlju detaljnije je opisano što je to automatsko postavljanje vremena, kao i načini na koje se ono može automatski postaviti. Izdvojena su dva modula, DCF77 i GPS (eng. *Global Positioning System*) te je definirana teorija koja stoji iza ta dva modula. U trećem poglavlju navodi se Arduino kao poveznica cijeloga rada i ostalo sklopovlje koje je korišteno poput RTC-a, GPS-a itd. Nakon definiranja pojedinog dijela te programske podrške slijedi četvrto poglavlje u kojem se čitav sklop testira. U zadnjem poglavlju rad je zaključen

1.1. Zadatak završnog rada

S arduino platformom potrebno je razviti digitlani sat s mogućnošću automatskog i ručnog podešavanja točnog vremena. Sat mora imati mogućnost prikaza vremena na matričnom pokazniku.

2. AUTOMATSKO POSTAVLJANJE VREMENA

Automatski postaviti vrijeme znači točno namjestiti vrijeme bez klasičnog namještanja vremena poput podešavanja kazaljki sata. To je ostvarivo uz korištenje određenih modula, prijamnika, koji primaju određeni signal iz nekog odašiljača te ga na adekvatan način pohranjuju i prikazuju. U sljedećim potpoglavljima sav postupak detaljnije je objašnjen.

2.1. Postavljanje točnog vremena

Prikazivanje vremena u svijetu može se podijeliti na dva glavna načina, a to su analogni prikaz i digitalni prikaz. [1]

Sl. 2.1.: Analogni sat.[1]

Sl. 2.2.: Alternativni dizajn analognog sata.[2]

Analogni prikaz podrazumijeva klasične zidne, odnosno ručne satove kod kojih se vrijeme prikazuje i mijenja pomicanjem kazaljki. Mogu biti upravljani mehanički i elektronički. Primjeri su dani na slikama 2.1. i 2.2. Za razliku od analognog prikaza, digitalni je jednostavniji za iščitavanje, uglavnom se koristi LCD i LED prikaz. Također mogu biti upravljani mehanički i elektronički. Primjeri takvih satova dani su na slikama 2.3. i 2.4.

Sl. 2.3. Digitalni sat.[3]

Sl. 2.4.: Binarni digitalni sat.[4]

[1] http://jadcotime.com.au/product-category/analogue-clocks/
[2] https://www.pinterest.com/pin/205195326751270215/
[3] https://www.masterclock.com/products/digital-clocks/
[4] http://www.hongkiat.com/blog/clock-designs/

Vrijeme se može postaviti ručno ili automatski. Sve više uređaja kojima je jedna od namjena prikazivanje vremena koriste automatsko podešavanje zbog točnosti. Također, automatsko postavljanje vremena ne zahtjeva nikakav dodatan rad korisnika sata. Uglavnom se radi o postavljanju točnog vremena po određenom atomskom satu (eng. *Atomic Clock*). Atomski sat temelji se na korištenju elektromagnetskih valova kako bi se dobilo točno vrijeme (točnost $7.24 * 10^{-9}\ s$)[5] [2].

U svrhu automatskog postavljanja vremena, u srednjoj Europi se često koristi DCF 77 sustav, a u svijetu postoje brojne njegove inačice koje rade na različitim frekvencijama poput WWVB-a u SAD-u te MSF-a u Ujedinjenom Kraljevstvu. Ipak, najrasprostranjeniji način automatskog podešavanja je onaj uz korištenje GPS modula. [3]

2.2. DCF 77

DCF 77 precizan je vremenski protokol koji se koristi u aplikacijama i sklopovima koji zahtijevaju automatsko podešavanje.

Neki od uređaja koji koriste DCF77 su:

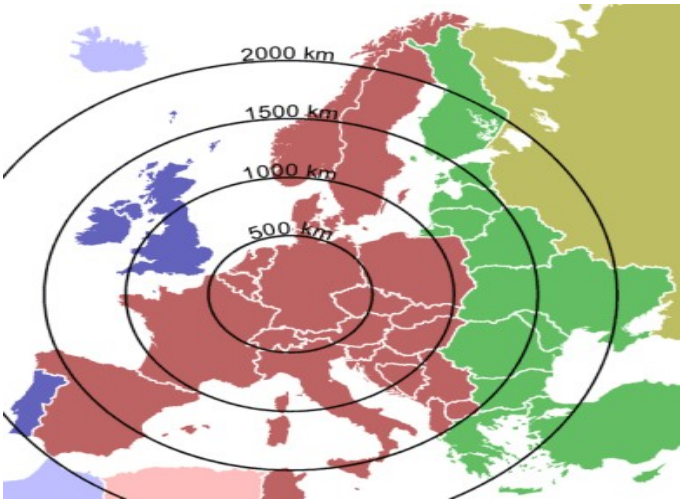
- satelitski prijemnici,
- alarmi,
- uređaji za pametno upravljanje električnom energijom.

Kratica DCF77 dolazi od:

- D – Njemačka (njem. *Deutschland*),
- C – dugovalni signal,
- F – Frankfurt,
- 77 – uređaj radi na 77.5 kHz.

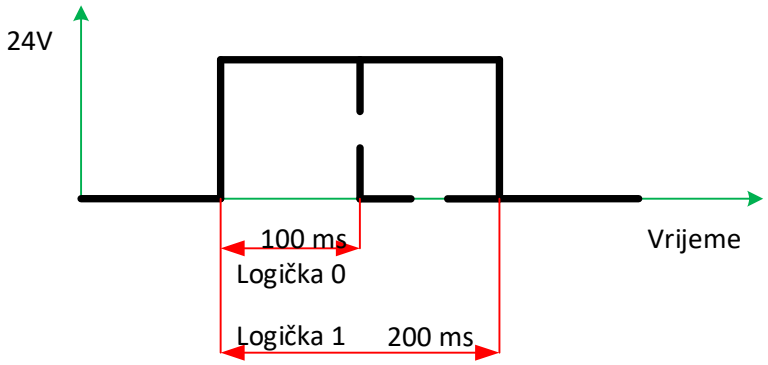
Signal se odašilje iz emitera kao amplitudno – modulirani signal koji sadrži točne informacije o trenutnom datumu i vremenu. Signal se šalje svake minute i precizan je $2 * 10^{-8}\ s$. Iako je odašiljač smješten u Frankfurtu, prijemnici koji su udaljeni i do 2000 km mogu primiti signal što se vidi na slici 2.5. [4]

[5]http://www.dailymail.co.uk/sciencetech/article-3607104/Is-second-change-Optical-atomic-clock-accurate-measure-time-don-t-worry-ll-never-notice-difference.html



Sl. 2.5.: Pojasi DCF 77 signala.[6]

Primljeni DCF 77 signal se sastoji od logičkih '0' i '1'. Nule se od jedinica razlikuju po duljini trajanja jednog pulsa, odnosno ako traje 100ms radi se o logičkoj 0, a ako traje 200 ms radi se o logičkoj 1. Zaprimljeni podaci su BCD (eng. *Binary Coded Decimal*) kodirani. Segment primljenog signala nakon pretvorbe izgleda kao na slici 2.6.

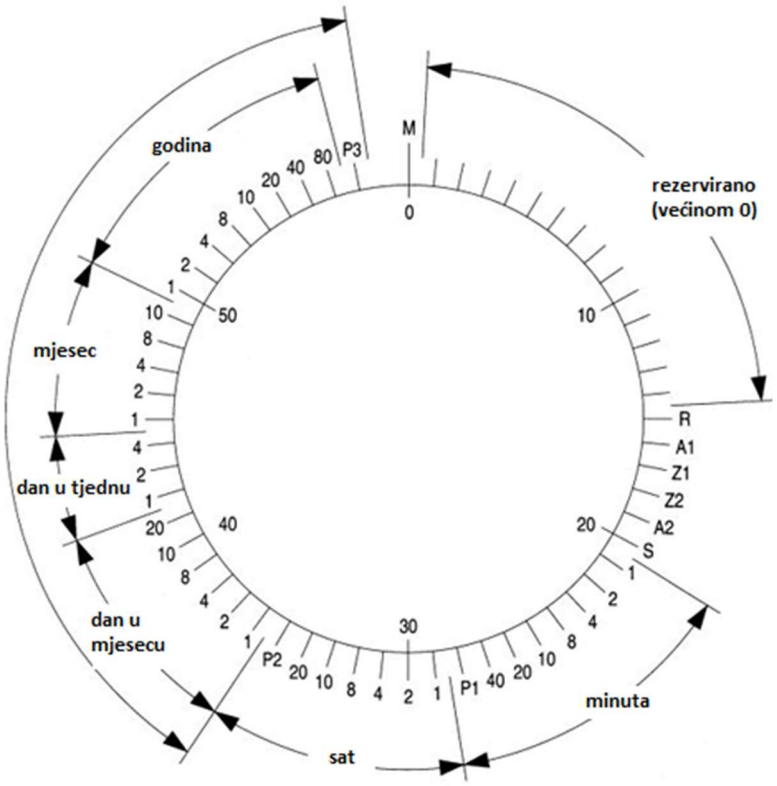


Sl. 2.6.: Signal nakon pretvorbe.

DCF77 signal može se primiti preko modula prikazanog na slici 2.7.

Sl. 2.7.: DCF modul.[7]

[6] https://www.compuphase.com/mp3/h0420_timecode.htm
[7] http://www.mathias-wilhelm.de/arduino/projects/pollin-dcf77-module/



Sl. 2.8.: Signal kodiran BCD kodom odašiljača DCF77.[8]

Prilikom dekodiranja modul očitava samo razlike u amplitudi frekvencije. DCF77 radio odašiljač smanjuje amplitudu svake sekunde, s iznimkom 59. sekunde jer tu sekundu koristi za sinkronizaciju toka podataka namijenjenu sljedećoj minuti. Amplituda je smanjena za 6 dB i pretvoreni signal onda izgleda kao na slici 2.6.

Kodirana poruka u BCD kodu i značenje te poruke prikazano je dijagramom na slici 2.8. Prvih 14 bitova se ignorira. Svi podaci šalju se za minutu koja slijedi. Postoje tri paritetna bita: onaj za minutu, za sat i za datum.

Značenje ostalih bitova je sljedeće:

- R – služi za kontrolu korištenja antene,
- A1 – najavljuje ljetno računanje vremena jedan sat unaprijed,
- Z1 – ima vrijednost logičke jedinice ako je ljetno računanje uključeno,
- Z2 – komplement Z1,
- A2 – najavljuje preskakanje sekunde jedan sat unaprijed,
- S – startni bit čija je vrijednost uvijek logička jedinica.

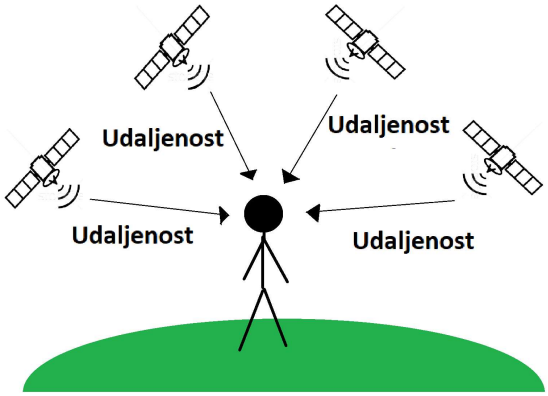
[8] https://www.compuphase.com/mp3/h0420_timecode.htm

Uz dane informacije signal se može dekodirati. Potrebno je promatrati podatke koje modul primi i zatim oduzeti sadašnje vrijednosti od vrijednosti prethodnog ciklusa. Tako možemo odrediti radi li se o logičkoj jedinici ili nuli. Poznato je da se radi o BCD kodu stoga je potrebno izvršiti dekodiranje i parsirati podatke.

2.3. GPS

GPS (eng. Global Positioning System) moduli nalaze se na uređajima koji zahtijevaju podatke kao što su lokacija, vrijeme i sl. Koriste sustave satelita koji kruže Zemljinom orbitom, kao i veliki broj stanica na Zemlji kako bi dali zahtijevane informacije. U svakom trenutku orbitira 24 satelita, a iz jedne točke najviše je vidljivo 12. Njihova svrha je odašiljati radio valovima informacije u stanice (Auto navigacija, pametni telefoni i sl.).

Podaci koje svaki od satelita pošalje sadrže različite informacije koje su nužne za točno određivanje lokacije, a za ovaj rad najbitnije, i vrijeme. Svaki satelit sadrži u sebi atomski sat. Informacije o vremenu i lokaciji satelita stižu s četiri različita satelita i zahvajujući tome može se izračunati udaljenost do bilo kojeg satelita vidljivog s određenog mjesta. Ako GPS antena može primiti podatke od najmanje četiri satelita, točno računa lokaciju i vrijeme (vidi Sl.2.9.) .Za završni rad korišten je GPS modul Neo 6M. Modul prima podatke u svom specifičnom formatu NMEA rečenice koji osim lokacije i vremena uključuju i još nekoliko korisnih podataka. [5]



Sl. 2.9.: GPS sustav.

3. SAT ZASNOVAN NA ARDUINO SUSTAVU

3.1. Arduino

Arduino je *open – source* platforma namijenjena za izgradnju elektroničkih projekata. Sastoji se od fizičke programibilne ploče (mikroupravljača) i dijela *software-a*, odnosno IDE (eng. *Integrated Development Environment*) koji se manipulira putem računala. Mikrokontroler se na računalo povezuje putem USB kabela.

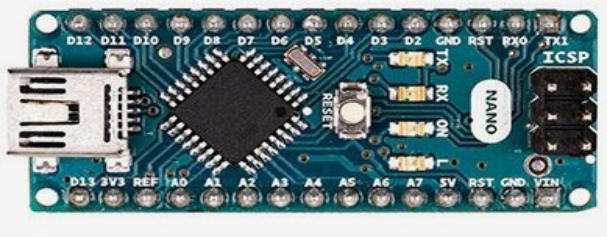
Jezik kojim se programira arduino podsjeća na C++, ali je pojednostavljen s ciljem da programerima olakša posao.

Arduino ima nekoliko inačica, a neke od njih su:

- Uno,
- Leonardo,
- 101,
- Micro,
- Mini,
- Nano.

U završnom radu korišten je Arduino Nano (vidi Sl. 3.1.) zbog svoje veličine i funkcionalnosti. Za razliku od Una, Nano koristi Mini - B USB kabel i pruža manju struju.

Jezgra većine Arduina je mikrokontroler Atmega328P s 32 KB memorije za spremanje koda. Serijskom komunikacijom (UART TTL 5V) povezan je s Arduino aplikacijom koja se koristi za programiranje. Sadrži 14 digitalnih nožica koji mogu koristiti kao ulazi (eng. *Input*) ili kao izlazi (eng. *Output*) i svaki od njih radi na naponu od 5V. Također, svaki pin može pružiti struju jakosti 40 mA i sadrži unutarnji *pull – up* otpornik od 20 do 50 kΩ. Osim 5V, Arduino ima pin koji pruža 3,3 V jer neki od modula zahtijevaju manji napon.



Sl. 3.1.: Arduino Nano.[9]

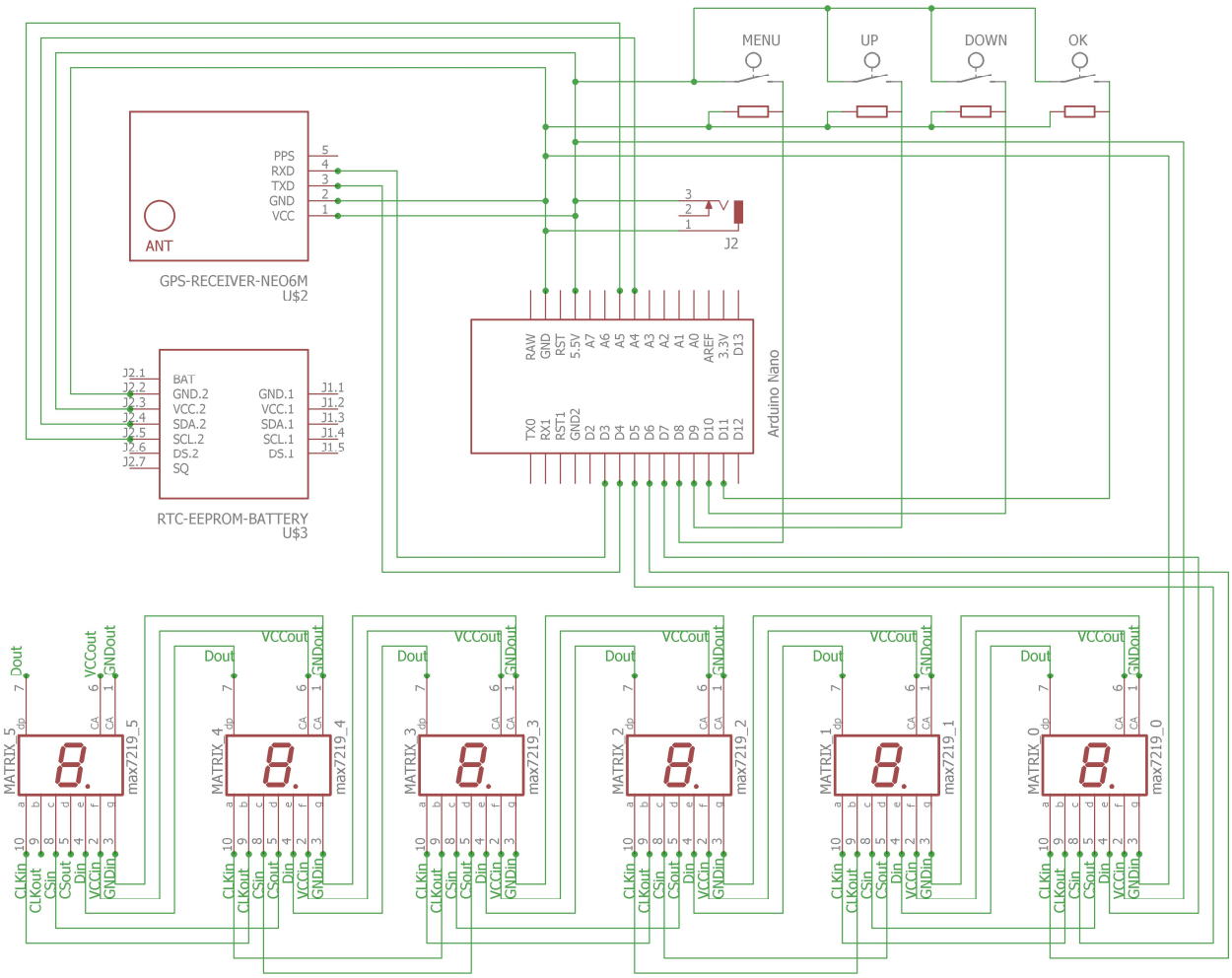
[9] https://store.arduino.cc/arduino-nano

Osim digitalnih nožica, Arduino sadrži i analogne pinove. Po tvorničkim postavkama i oni rade na 5V, no moguće je, manipuliranjem koda, smanjiti taj napon. [6]

U završnom radu korištene su digitalne nožice, a sve je detaljnije objašnjeno u sljedećim poglavljima.

3.2. Sklopovlje

Na slici 3.2. prikazana je konačna shema razvijenog sata projektirana u programskom okruženju Eagle. Slika je u narednim potpoglavljima detaljnije objašnjena.



Sl. 3.2.: Shema sklopovlja.

3.2.1. RTC

Real Time Clock je uređaj koji vodi brigu o održavanju vremena. Nalazi se na svakom računalu gdje mu je, kao i u ovom radu, glavna svrha pohranjivanje vremena kada računalo nije pokrenuto odnosno kada električna energija nije dostupna. Iako se spremanje vremena može postići i drugim sredstvima, korištenje RTC-a ima više prednosti kao što su:

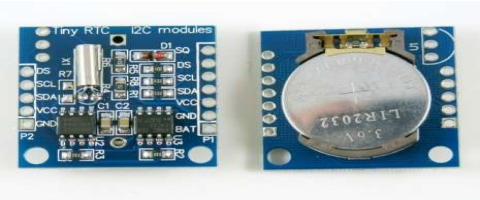
- RTC troši manje struje,
- često je točniji od ostalih načina (brojači mikroupravljača, programska rješenja)
- glavni sustav je oslobođen na korist važnijih zadataka.

Većina koristi bateriju kao izvor napajanja za pohranu vremena. Osim što pohranjuje vrijeme, RTC isto vrijeme osvježava u stabilnom vremenskom periodu (npr. svaku sekundu). Da isto bilo moguće potrebna je stabilna frekvencija osciliranja. Zato svaki RTC sadrži oscilator koji kod većine titra nominalnom frekvencijom 32.768 kHz što je i slučaj u ovom radu. Vanjski faktori poput naglih temperaturnih promjena mogu utjecati na frekvenciju, a samim time i na točnost vremena.

U ovom radu korišten je DS1307 RTC koji troši vrlo malo energije i osim mogućnosti održavanja vremena za sat, može i kalendarski pratiti vrijeme. Podatke sprema u 56 bajta NV SRAM-a (eng. *Non Volatile Static Random Access Memory*) gdje su podaci zapisani u BCD kodu. Ima ugrađen senzor za očitanje gubitka energije koji omogućava automatsko prebacivanje na baterijski izvor napajanje te nastavak rada. [7]

Na slici 3.3 lijevo prikazana je donja strana modula. Uočljivo je sedam nožica (uključujući baterijski) od kojih se u radu koristi samo 4, a način povezivanja vidljiv je na shematskom prikazu rada.

Nožice VCC i GND koriste se za dovod napona, odnosno uzemljenje modula. Nožica SDA (eng. *Serial Data Input/Output*) služi za ulaz i izlaz podataka serijskog I2C (eng. *Inter – Integrated Circuit*) sučelja, a povezan je s arduinom I2C nožicama (A4 i A5). Nožica SCL (eng. *Serial Clock Input*) služi za sinkronizaciju, odnosno podešavanje protoka podataka serijskim sučeljem.



Sl. 3.3 : RTC DS1307.[10]

[10] http://hub360.com.ng/shop-2/ds1307-rtc-module/

3.2.2. GPS

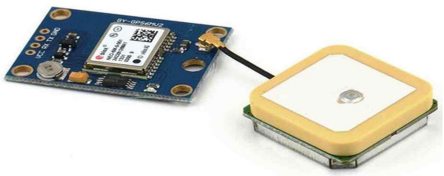
Način na koji GPS općenito funkcionira objašnjen je u poglavlju 2.3. GPS. U ovom poglavlju detaljnije je objašnjen modul koji se koristi u ovom radu. Korišteni modul proizvođača Ublox poznat je pod nazivom Neo 6M, a prikazan na slici 3.4. U ovom radu se koristi samo jedna od njegovih mnogobrojnih mogućnosti, a to je sinkronizacija sata vrijednostima atomskog sata na GPS satelitu.

Prijem signala omogućava pripadajuća antena. Antena modula može se podijeliti na dva dijela: aktivni i pasivni. Pasivna antena može primiti signal i poslati ga kroz kabel do sklopa, a aktivna taj signal pojačava da bi neutralizirala određeni dio vanjskih smetnji. Antena je na čip povezana koaksijalnim kablom koji dodatno sprječava vanjske smetnje.

Podatke primljene putem antene zovemo NMEA rečenica (eng. *National Marine Electronics Association*). Takvu rečenicu GPS putem serijske komunikacije prosljeđuje Arduinu koji, korištenjem biblioteke rečenicu dešifrira i pretvara u oblik koji se dalje koristi. Serijska komunikacija (UART – eng. *Universal Asynchronous Receiver/Transmitter*) može biti sklopovska (bolji izbor) i programski realizirana. UART se koristi npr. kod dial-up modema koji se spajaju na matičnu ploču kao i kod ostalog sklopovlja koje se spaja na isti način. U radu je korištena softverska komunikacija Arduino nožicama 3 i 4 od kojih je nožica modula RX spojen na Arduino nožicu 3 i služi za ulaz , a TX je spojen na Arduino nožicu 4 i služi za izlaz. Obje nožice koriste TTL (eng. *Transistor Transistor Logic*) logičke razine. [8]

Primjer NMEA rečenice - $GPVTG,0.00,T,,M,0.00,N,0.00,K,N*31

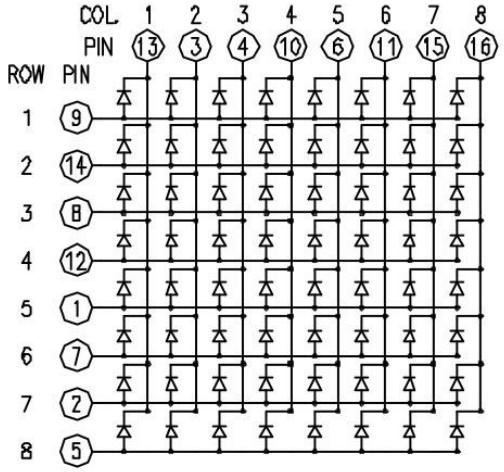
Osim te dvije nožice, kao i svaki drugi modul, GPS prima napon 5 V kroz nožicu VCC, dok je uzemljen pomoću GND.



Sl. 3.4.: GPS Neo 6M.[11]

[11] https://robu.in/product/ublox-neo-6m-gps-module/

3.2.3. 8x8 LED Matrix



Sl. 3.5.: Shema LED matrice.[12]

LED Matrix je skup od 64 svijetleće diode koje mogu biti povezane u recima zajedničkom anodom, a u stupcima zajedničkom katodom ili obrnuto (vidi Sl. 3.5.). U radu se koriste diode crvene boje koje su cijenom najjeftinije. Skuplja varijanta bila bi RGB (eng. *Red Green Blue*) koja bi omogućila promjenu boja.

U ovom slučaju stupci su povezani na katode dioda, a reci na anode. Zbog toga stanje stupca mora biti logička nula da bi bilo koja dioda u tome stupcu svijetlila, te stanje retka mora biti logička jedinica da bi odabrana dioda svijetlila. Ukoliko su i stupac i redak odabrane diode u stanju logičke nule ili logičke jedinice, dioda neće svijetliti. Za upravljanje većim brojem primjenjuje se slična tehnika. Redak mora biti postavljen u jedinicu, dok se stupci postavljaju ovisno u jedinicu ili nulu ovisno o tome koju diodu u redu želimo upaliti, a koju ne. Kontroliranje dioda na ovaj način oduzelo bi mnogo vremena stoga je u radu korišten integrirani sklop MAX7219. [9]

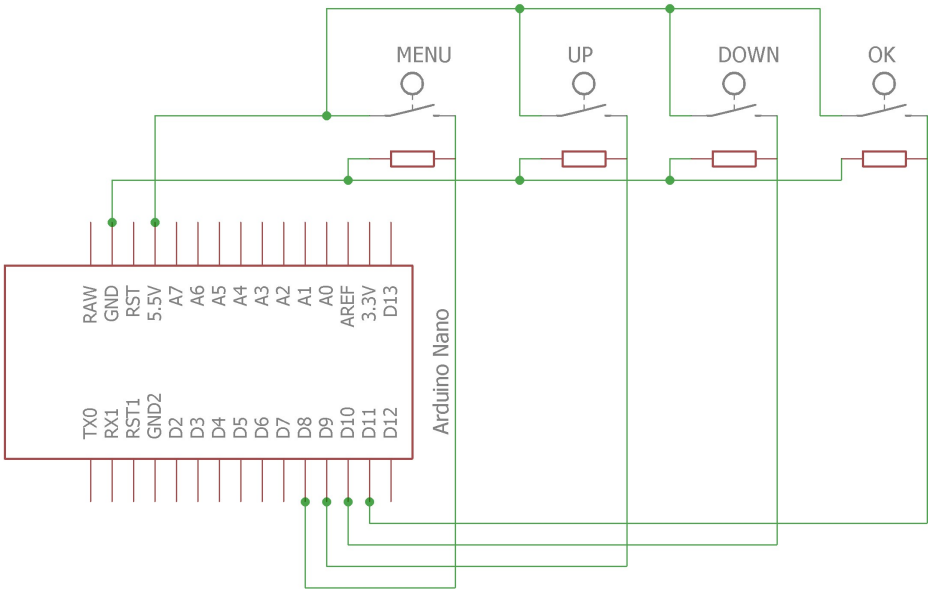
MAX7219 je kompaktan serijski ulazno – izlazni upravljač LED zaslonima ili, u ovom slučaju, LED matricama. Sklop u sebi sadrži BCD koder – dekoder, multiplekser, upravljački program za gore navedene pokazivače i 8x8 statički RAM za pohranjivanje pojedine znamenke. Nožicama VCC i GND sklop je povezan na napon od 5V, odnosno uzemljen. Nožice DIN, CS i CLK povezane su na digitalne nožice arduina. Ukoliko se koristi više matrica kao u ovome radu, upravljački programi vrlo se jednostavno mogu proslijediti ostalim MAX7219 nožicom DIN (eng. *Data IN*), odnosno DOUT (eng. *Data OUT*). Osim DIN i DOUT tu su i ostale dvije nožice, CS

[12] https://www.arduino.cc/en/Tutorial/RowColumnScanning

(eng. *Chip Select*) te CLK (eng. *Clock Pulse Source*) koji se spajaju s izlaza jedne matrice na ulaz druge, odnosno u početku s digitalne nožice na nožicu povezanu s čipom. [10]

Manipuliranje radom sklopa u ovom je radu postignuto korištenjem biblioteke koja funkcionira na temelju metode zajeničke anode/katode već objašnjene u ovom poglavlju.

3.2.4. Tipkala



Sl. 3.6.: Shema spajanja tipkala.

Tipkalo je elektronička komponenta koja pritiskom spaja dvije točke u strujnom krugu. U ovom radu korištena su 4 tipkala čija je svrha upravljanje menijem. Tipkalo *MENI* služi samo za pristup meniju, tipkala *gore* i *dolje* služe za navigaciju podmenijima, a posljednja tipka *OK* služi za potvrdu odabira. Tipkala se spaju na arduino kao na slici 3.6.

3.3. Programska podrška

Na početku koda deklarirane su biblioteke, objekti i varijable koji će se koristiti. Varijablama je dodjeljena početna vrijednost. Neke od njih bit će korištene kao oznake za nožice radi lakšeg snalaženja u kodu.

U funkciji setup određeno je da će se serijska komunikacija odvijati brzinom 9600 bauda. LE dioda na nožici 13 deklarirana je kao izlaz. Nakon toga se odvija inicijalizacija matrica prikazana

Linija ***Kod***

```
1:  void initMatrix()
2:  {
3:    for (int i=0; i< NBR_MTX; i++){
4:      max7219.shutdown(i,false);
5:      max7219.setIntensity(i,5);
6:      max7219.clearDisplay(i);
7:       }
8:   max7219.clearAll();
9:  }
```

Sl. 3.7.: Inicijalizacija matrica.

na slici 3.7. Matrice prvo zasvijetle, zatim se postavlja razina osvjetljenja i na kraju se sve diode prestanu svijetliti te su spremne za daljnje korištenje.

Nakon matrica, nožice predviđene za tipkala deklariraju se kao ulazi i uz korištenje biblioteke Bounce2 na istima je primijenjen debounce.

Biblioteka Bounce2 provjerava promjenu stanja nožice svakih 10 ms (u ovom slučaju). Period u kojem provjerava dobiva pomoću funkcije *millis()* koja kao rezultat vraća broj milisekundi koji je protekao od početka rada trenutnog programa. Postoji i varijanta s korištenjem funkcije *delay()*, no ne preporučuje se zbog toga što *delay()* pauzira sve dok ne završi period. U biblioteci su korištene varijable tipa bool koje služe kao zastavice za naznačavanje trenutnog i prošlog stanja. Također, moguće je birati uzima li biblioteka u obzir rastući ili padajući brid.

Osim toga, u funkciji setup podaci iz memorije EEPROM (eng. *Electricaly Erasable Programmable Read-Only Memory*) se pohranjuju u varijablu koja će se kasnije koristiti. EEPROM je memorija u kojoj podaci ostaju i nakon što se arduino odspoji s izvora napajanja. Veličina te memorije iznosi 1 KB.

U funkciji loop pozivaju se razvijene funkcije. Za razliku od funkcije setup koja se obavi samo jednom, na početku programa, funkcija *loop()* odvija se neprekidno, osim ako nije programski prekinuta. Ista se nalazi na slici 3.8. Naredba u trećoj liniji služi za provjeru tipkala, odnosno pripremu za moguće uključenje tipkala. Ako je logičko stanje tipkala bilo promjenjeno logičke varijable u biblioteci se ažuriraju. Arduino stalno provjerava je li GPS modul uspostavio vezu. Zbog kristalnog oscilatora točno svake sekunde parsira podatke, koji su zapravo NMEA rečenica. Uvjet za sve predstavlja varijabla fixCount koja služi za automatsko sinkroniziranje vremena svakih 45 minuta. Pravilnost podataka ispituje se logičkim varijablama. Provjerava se pravilnost podataka, sadrže li podaci vrijeme i jesu li pravovremeni ili su iz unutarnjeg RTC-a GPS-a, a provjera se vidi u osmoj liniji slike 3.8.

Linija ***Kod***

```
1:  void loop()
2:  {
3:    debouncerMenu.update();
4:    if (gps.available( gpsPort ))
5:       {
6:      gps_fix fix = gps.read();
7:      fixCount++;
          if (fix.valid.time && fix.valid.status && (fix.status >=
8:  gps_fix::STATUS_STD))
9:      {
10:           if (fixCount - lastGPSsync >= GPS_SYNC_INTERVAL)
11:           {
12:             clockElements.Hour   = fix.dateTime.hours;
13:             clockElements.Minute = fix.dateTime.minutes;
14:             clockElements.Second = fix.dateTime.seconds;
15:             lastGPSsync = fixCount;
16:             RTC.write( clockElements );
17:             digitalWrite( LED_PIN, !digitalRead(LED_PIN) );
18:           }
19:         }
20:         RTC.read( clockElements );
21:      printData();
22:    }
23:    if (debouncerMenu.rose() )
24:    {
25:      mainMenu = 0;
26:      menuProcedure();
27:    }
28:  }
```

Sl. 3.8.: Glavna funkcija - loop.

Podaci su dobiveni parsiranjem NMEA rečenice korištenjem biblioteke NMEAGPS.h. Ako je rezultat provjera u osmoj liniji logička jedinica još se ispituje je li prošlo barem 45 minuta od poslijednje provjere. Ako jest, trenutno vrijeme sprema se u *fix* podatkovnu strukturu u cjelobrojnom obliku, nakon čega se cijela *fix* struktura sprema u vremensku strukturu clockElements u obliku bajtnom obliku. Takva struktura predaje se RTC-u na pohranu.

U slučaju da barem jedan od uvjeta nije zadovoljen, umjesto pohranjivanja vremena u RTC, stari podaci se čitaju iz RTC-a. Podaci se spremaju u istu strukturu *clockElements* koja se ispisuje pozivom na funkciju *printData()* vidljivoj na slici 3.9. Naposljetku, svaki ciklus provjerava se je li tipka za pristup meniju pritisnuta. Ako je, pokreće se procedura prikaza menija.

Prije poziva funkcije *printData()*, podaci nisu prilagođeni vremenskoj zoni u kojoj se nalazi krajnji korisnik. Potrebno je odabrati željenu vremensku zonu navigiranjem kroz meni koji je

opisan u nastavku teksta. Ako je vrijednost EEPROM zastavice postavljena u logičku jedinicu znači da je vremenska zona promijenjena i potrebno podatke prilagoditi istoj. Prethodno prilagođeni podaci prvo su prebačeni u vremensku zonu nula, nakon čega im je dodana odnosno oduzeta određena količina sati koja odgovara odabranoj vremenskoj zoni.

Linija ***Kod***

```
1:  void printData(){
2:    if (eepromFlag == true)
3:    {
4:         if( previousEepromZone >= 11)
5:      {
6:        var1 = previousEepromZone - 11;
7:           clockElements.Hour = clockElements.Hour - var1;
8:      }
9:         else if ( previousEepromZone < 12 )
10:          clockElements.Hour = clockElements.Hour + previousEepromZone;
11:        if ( clockElements.Hour > 200)
12:        {
13:          clockElements.Hour = 256 - clockElements.Hour;
14:          clockElements.Hour = 24 - clockElements.Hour;
15:        }
16:        clockElements.Hour = clockElements.Hour % 24;
17:        eepromFlag = false;
18:      }
19:        if( eepromZone >= 11)
20:     {
21:       var2 = eepromZone - 11;
22:       clockElements.Hour = clockElements.Hour + var2;
23:     }
24:     else if ( eepromZone < 12 )
25:       clockElements.Hour = clockElements.Hour - eepromZone;
26:     if ( clockElements.Hour > 200)
27:     {
28:       clockElements.Hour = 256 - clockElements.Hour;
29:       clockElements.Hour = 24 - clockElements.Hour;
30:     }
31:     clockElements.Hour = clockElements.Hour % 24;
32:     max7219.setChar(0, clockElements.Second % 10);
33:     max7219.setChar(1, clockElements.Second / 10);
34:     max7219.setChar(2, clockElements.Minute % 10);
35:     max7219.setChar(3, clockElements.Minute / 10);
36:     max7219.setChar(4, clockElements.Hour % 10);
37:     max7219.setChar(5, clockElements.Hour / 10);
38: }
```

Sl. 3.9.: Funkcija za ispis podataka.

Spremljeni podaci tipa su bajt što znači da nema negativnih brojeva već nakon nule slijedi broj 255. Zbog toga je od ukupnog broja brojeva, 256, oduzet broj sati, ali samo u slučaju ako je broj

manji od nule. Novodobiveni broj potrebno je tada oduzeti od 24 kako bi bio u 24 satnom formatu. U slučaju da je broj veći od nule potrebno je uzeti u obzir mogućnost da prilikom zbroja broj prijeđe 24. Zbog tog slučaja uzet je ostatak prilikom cjelobrojnog djeljenja broja 24 s dobivenim brojem sati.

U tekstu iznad navedeno je da ako je tipka meni pritisnuta pokreće se procedura za pristup meniju. Prilikom ulaska u meni ponuđene su četiri mogućnosti: prisilna sinkronizacija, ručno postavljanje vremena, odabir vremenske zone i izlaz i povratak u glavnu petlju.

Prisilna sinkronizacija radi tako što se kontrolne varijable postave na odgovarajuće vrijednosti. Varijabla za brojanje sekundi, odnosno za brojanje 45 minuta do sljedeće sinkronizacije postavlja se na broj 2698. To je dvije sekunde prije 45 minuta što daje dovoljno vremena GPS-u da preda barem jednu cijelu NMEA rečenicu. Također, kontrolna varijabla za broj sinkronizacija postavlja se u nulu.

Vrijeme se ručno postavlja korištenjem tipkala. Tipkalo *gore* služi za povećavanje broja sati, a kada taj broj bude 23, resetira se i brojanje kreće ispočetka. Isti princip primijenjen je i za minute osim što se koristi drugo tipkalo (tipkalo *dolje*). Pritiskom na tipkalo *OK* vrijeme se pohranjuje u RTC i koristi se sve do sljedeće sinkronizacije. Važno je napomenuti da se kontrolne varijable za sinkronizaciju resetiraju.

Linija	***Kod***
1:	`if (debouncerOK.rose())`
2:	`{`
3:	`  if ( currentZone < 0)`
4:	`  currentZone = currentZone * (-1);`
5:	`  else`
6:	`  currentZone = currentZone + 11;`
7:	`  EEPROM.write (0, currentZone);`
8:	`  previousEepromZone = eepromZone;`
9:	`  eepromZone = currentZone;`
10:	`  eepromFlag = true;`
11:	`  break;`

Sl. 3.10.:Izlaz iz podmenija odabir zone.

Promjena vremenske zone također se izvodi korištenjem tipkala. Na početku je potrebno pročitati podatke iz RTC koji su karakteristični za nultu vremensku zonu. Postoji 24 vremenske zone i moguće je odabrati bilo koju od njih. Prilikom odabira omogućen je pretpregled zona gdje su samo sati i minute, a umjesto sekunda naznačeno je koja vremenska zona je trenutno odabrana.

Sati se zbrajaju, odnosno oduzimaju od postojećih za svaku zonu posebno od sati u nultoj vremenskoj zoni. Nakon što su dodani ili oduzeti javlja se isti problem kao i kod funkcije za

prikazivanje vremena. Pošto je tip podataka bajt potrebno je ponoviti cijeli postupak jer se radi o drugoj vremenskoj varijabli. Nakon odrađenog postupka poziva se funkcija za ispis, odnosno prikaz vremena zone u meniju. Ako je na nekom meniju pritisnuto tipkalo *OK* arduino zapisuje u EEPROM memoriju o kojoj se vremenskoj zoni radi tako da taj podatak bude dostupan i nakon isključivanja i ponovnog uključivanja sklopa u struju (Sl. 3.10.). Ako se radi o negativnim zonama, uzima se njihova apsolutna vrijednost, a ako se radi o pozitivnim zonama na njihovu vrijednost dodaje se broj 11 koji se prilikom čitanja oduzima.

Pisanje i čitanje iz RTC-a odvija se preko I2C sučelja, odnosno biblioteke *Wire*. I2C sučelje je serijski protokol koji omogućuje spajanje više uređaja, modula uz brzinu prijenosa podataka koja odgovara aplikaciji. [11]

Linija	***Kod***
1:	`debouncerUp.update();`
2:	`debouncerDown.update();`
3:	`debouncerOK.update();`
4:	`if (debouncerUp.rose())`
5:	`{`
6:	`mainMenu++;  //Going up the menu`
7:	`}`
8:	`if (debouncerDown.rose())`
9:	`{`
10:	`mainMenu--; //Going down the menu`
11:	`}`
12:	`if (mainMenu == 0)`
13:	`{`
14:	`Menu0(); //Shows "MENU", has no real function`
15:	`}`

Sl. 3.11.: Navigiranje menijem.

Upravljanje menijem temeljeno je na jednoj varijabli. Ona se povećava ili smanjuje ovisno o tome koje tipkalo je pritisnuto, a vrijednost označava koji meni se prikazuje (vidi Sl. 3.11.).

4. TESTIRANJE

Sl. 4.1.: Sklop s kućištem.

Nakon završetka pisanja koda cijeli sklop je testiran. Zbog estetskih razloga kao i zbog olakšanog prijenosa izrađeno je kućište za sklop. Materijal korišten za kućište je pleksiglas. Rad je vidljiv na slici 4.1.

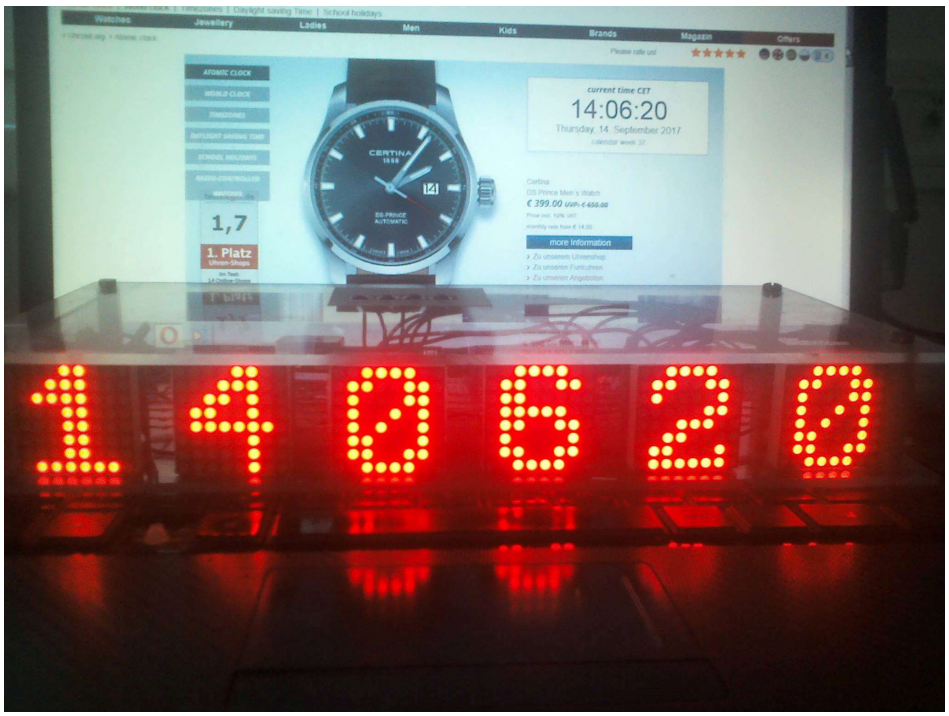
Kada GPS modul ostvari vezu sa satelitom LE dioda koja se nalazi na njemu zasvijetli plavom bojom. Period između svakog početka i kraja svijetljenja iste diode točno je sekundu i sinkroniziran je s kristalnim oscilatorom na modulu koji je spomenut u poglavlju 3.3.

Sl. 4.2.: Sklop s kućištem uz stabilnu vezu sa satelitom i sinkronizacijom.

Također, na arduinu se nalazi jedna LE dioda na digitalnom pinu 13. U svrhu testiranja dodana je mogućnost da zasvijetli svaki put kada se RTC sinkronizira GPS vremenom (svakih 45 minuta ili prisilnim sinkroniziranjem). Taj slučaj vidljiv je na slici 4.2.

Nakon pet dana rada bez sinkronizacije s GPS-om (odspojena antena) vrijeme pohranjeno u RTC-u bilo je jednu minutu više u usporedbi s točnim GPS vremenom. Pretpostavljeni razlog je velika razlika u temperaturi tih dana (noći puno hladnije od dana). Također je poznato da korišteni modul nema mogućnost sprege temperaturnih razlika. Precizniji modul s tom mogućnošću može se naći po skupljoj cijeni i pod nazivom DS3231. [12]

Nakon što je donesen zaključak da je vrijeme u RTC-u jednu minutu žurilo, antena je spojena s GPS-om i pokrenuta je funkcija prisilne sinkronizacije. Na slici 4.3. vidljivo je kako je vrijeme opet točno



Sl. 4.3.: Usporedba vremena sklopa s vremenom na web stranici koja se sinkronizira s atomskim satom[13].

[13] https://watches.uhrzeit.org/atomic-clock.php

5. ZAKLJUČAK

U završnom radu razvijen je digitalni sat s mogućnošću automatskog i ručnog postavljanja točnog vremena. Prikaz vremena odvija se na šest matričnih pokaznika. U radu su objašnjene osnove arduina kao razvojnog sustava, navedeni su i objašnjeni svi sklopovi koji su korišteni.

Dani su važni dijelovi koda koji su također objašnjeni. Testiranje je dovelo do zaključka kako vrijeme RTC-a treba barem jednom dnevno sinkronizirati GPS-om kako bi vrijeme u bilo kojem trenutku kada je potrebno bilo točno. Nakon pet dana rada bez sinkronizacije s GPS-om ura je pokazivala jednu minutu više što znači da korišteni RTC modul nije dovoljno točan.

Prilikom testiranja matrični pokaznici preskočili su sekundu, odnosno ne prikazati je iako je ona prošla u RTC-u. Problem se javlja vrlo rijetko (tri puta u pet minuta), no moguće ga je otkloniti korištenjem prekida (ISR - eng. *Interrupt Service Routine*) koja bi prekinula izvođenje programa svakih 45 minuta u svrhu sinkronizacije vremena. Taj način rezultirao bi slabijom potrošnjom memorije, a arduino ne bi svake sekunde provjeravao postoji li veza sa satelitom.

U konačnici, rad prikazuje točno vrijeme uz zanemarive nepravilnosti te je ideja uspješno implementirana.

LITERATURA

[1] Analogno i digitalno, http://www.dummies.com/programming/electronics/the-difference-between-analog-and-digital-electronics/, pristupljeno 6. rujna 2017.

[2] Atomski sat, http://tf.nist.gov/general/pdf/2039.pdf 28. lipnja 2017.

[3] DCF77, MSF, WWVB, https://www.cl.cam.ac.uk/~mgk25/time/lf-clocks/ , pristupljeno 5. rujna 2017.

[4] DCF77, http://tf.nist.gov/general/pdf/2429.pdf, pristupljeno 5. rujna 2017.

[5] GPS, http://www.trimble.com/gps_tutorial/howgps-timing.aspx, pristupljeno 28. lipnja 2017.

[6] Arduino, https://www.arduino.cc/en/Guide/Introduction, pristupljeno 29. lipnja 2017.

[7] DS 1307, https://datasheets.maximintegrated.com/en/ds/DS1307.pdf, pristupljeno 5. rujna 2017.

[8] GPS Neo 6M, https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf, pristupljeno 5. rujna 2017.

[9] 8x8 matrice, https://www.arduino.cc/en/Tutorial/RowColumnScanning, pristupljeno 5. rujna 2017.

[10] MAX 7219, https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf, pristupljeno 5. rujna 2017.

[11] I2C, http://www.glacialwanderer.com/hobbyrobotics/?p=12, pristupljeno 9. rujna 2017.

[12] DS3231, https://datasheets.maximintegrated.com/en/ds/DS3231.pdf, pristupljeno 14. rujna 2017.

SAŽETAK

Cilj ovog rada bio je projektirati funkcionalni digitalni sat s mogućnošću automatskog podešavanja. Prije svega bilo je potrebno pronaći odgovarajuće sklopovlje te sve povezati preko arduino platforme. Arduino je platforma namijenjena za izgradnju elektroničkih projekata. Za preuzimanje točnog vremena sa satelita korišten je GPS Neo 6M modul. Za pohranjivanje i osvježavanje tog vremena bio je potreban RTC, a odabran je modul DS1307. Podatke je bilo još potrebno prikazati, a to je bilo potrebno napraviti pomoću matričnih pokaznika. Odabran je dvadesetčetverosatni prikaz gdje su po dva matrična pokaznika prikazivala po jedan segment (sat, minuta, sekunda). Na kraju je programski ostvaren i meni zbog lakšeg pristupa opcijama poput odabira zone, ručnog postavljanja i prisilne sinkronizacije. Korištena su četiri tipkala koja su bila dostatna za navigaciju menijem. Nakon što je sve bilo završeno cijeli sklop bilo je potrebno zaštititi. Napravljeno je kućište od pleksiglasa.

Ključne riječi: arduino, GPS, RTC, matrice.

ABSTRACT

The goal of this paper was to develop an automatically synchronized watch. Firstly, it was necessary to obtain all the needed hardware and connect it to Arduino. Arduino is a platform with a purpose of building electronical projects. GPS Neo 6M module was used to get the needed time information from satellite. Storing and updating the time from GPS was task of RTC module DS1307. All the data obtained had to be shown on 8x8 matrices. The format used to display the time was 24 hour format and each time segment (hour, minute, second) got 2 matrices. In the end, the menu was created so it could be easier to access different options such as zone choosing, manual time setting and force time synchronization. Cruising through the menu was made possible with only four buttons. After all that was finished, hardware needed to be protected by making a case. The case was made from plexiglass.

Key words: Arduino, GPS, RTC, matrices.

ŽIVOTOPIS

Filip Begić rođen je 13. Studenoga 1995. Godine u Požegi, gdje trenutno i živi. Završio je Osnovnu školu Antuna Kanižlića u Požegi, nakon čega upisuje Gimnaziju u Požegi, smjer Prirodoslovno – matematički koju završava 2014. godine. Iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

P1 Programska realizacija nalazi se u .ino datoteci koja se otvara pomoću programa Arduino.

P2 Shema čitavog sklopovlja nalazi se u .sch datoteci koja se otvara pomoću programa EAGLE.