TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA

GETALDIĆEVA 4, ZAGREB

ZAVRŠNI STRUČNI RAD:

Digitalni sat ispisan riječima

MENTOR: Ivan Jurić UČENIK: Marko Zaninović

RAZRED: 4.B

U Zagrebu, 22. travnja 2019.

**SADRŽAJ**

1. Projekt Arduino
   1. Uvod………………………………………………………………..str1
   2. Povijest……………………………………………………………..str2
2. Hardware Arduina
   1. Uvod………………………………………………………………..str3
   2. Arduino UNO………………………………………………………str4
3. Software
   1. Arduino IDE………………………………………………………..str6
   2. Struktura Sketcha …………………………………………………. Str7
4. Izradba digitalnog sata ispisanog riječima
   1. Komponente……………………………………………………….. str8
   2. Elektronički spoj…………………………………………………... str12
   3. Objašnjen programski kod……………………………………….... str14
5. Programski kod sata………………………………………………….… str19
6. Zaključak……………………………………………………………….. str24
7. Literatura……………………………………………………………….. str25

**1. Projekt Arduino**

**1.1 Uvod**

Arduino je otvorena (open-source) računalna i softverska tvrtka koja dizajnira mikrokontrolere za građu digitalnih uređaja i naprava. On omogućuje jednostavno spajanje računala sa fizičkim svijetom. Arduino je licenciran od strane „GNU Lesser General Public License“, što omogućava svima da sastavljaju vlastite Arduino pločice, kao i Arduino softver. Komercijalno se Arduino pločice prodaju samostalno ili u sastavu „Do it yourself“ kompleta.

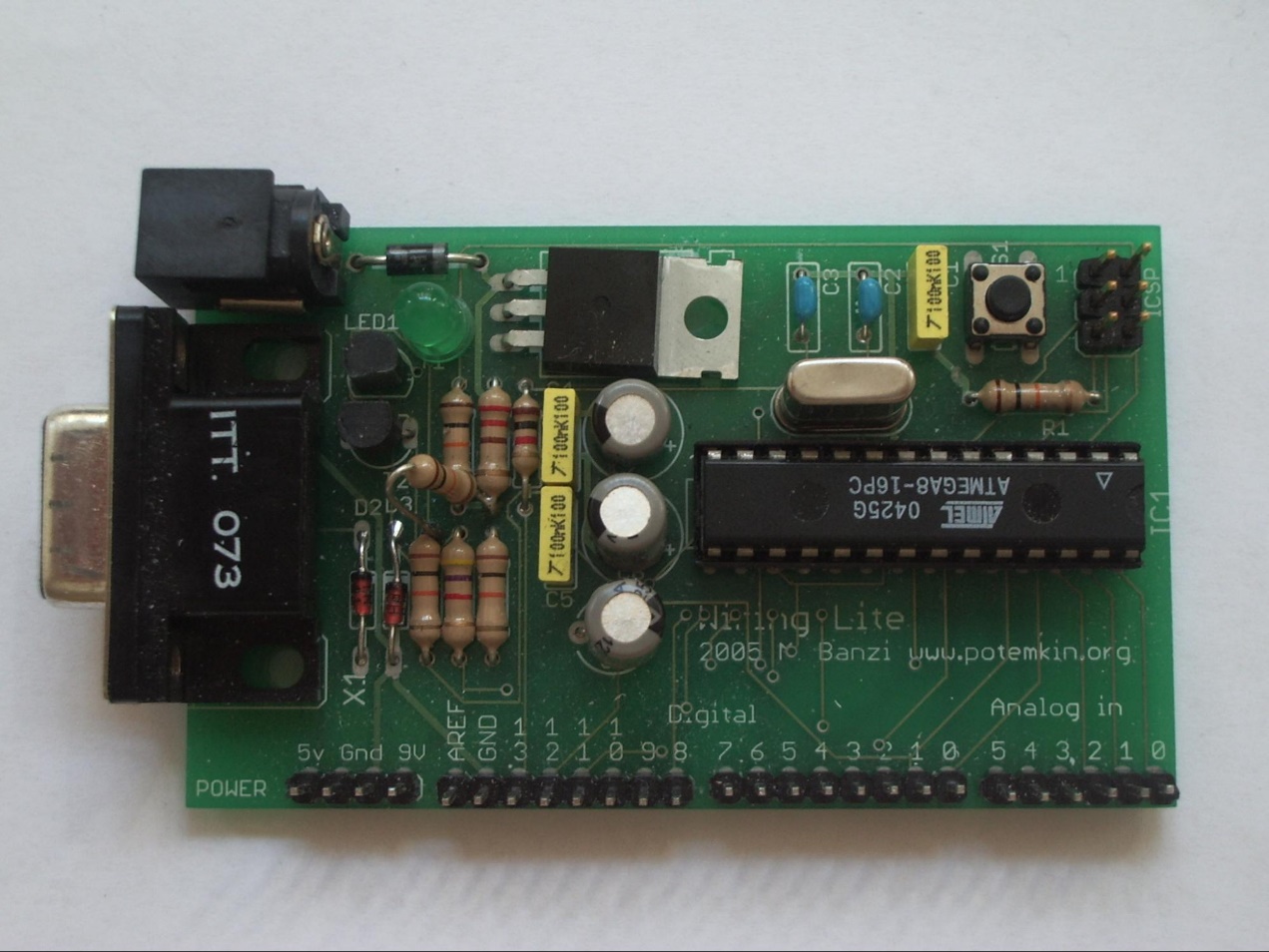
Slika 1. Arduino logo

Jedan od ciljeva Arduina je jednostavnost, odnosno mogućnost svakoga da kupi komplet i počne stvarati vlastite projekte. Zbog toga Arduino je najpopularnija platforma za stvaranje projekata. Koriste ga amateri, umjetnici, profesionalci i inženjeri. Također je zbog aspekta jednostavnosti često korišten za akademske svrhe, jer je Arduino sjajan uvod u svijet mikrokontrolera i programiranja.

**1.2 Povijest**

Projekt Arduino započeli su Casey Reas i Massimo Banzi 2003. godine u obliku studentskog programa u Italiji. Cilj je bio stvoriti jeftin i lagan način kako bi studenti i profesionalci mogli stvarati uređaje koji bi komunicirali sa svijetom oko njih, koristeći senzore i aktivatore. Česti primjeri takvih uređaja su roboti, termostati i detektori pokreta.

Prvi model Arduina sastojao se od PCB pločice, ATmega168 mikrokontrolera i jednostavnih biblioteka koje su omogućavale njegovo programiranje. Kako bi još snizili cijenu pločice, dodali su potporu za ATmega8 mikrokontroler.

Danas se Projekt Arduino sastoji od dvije tvrtke: Arduino LLC, koji se brine o softverskom razvoju projekta, i od Arduino SRL, koji sastavlja i razvija PCB pločice. U njihovoj povijesti nalaze se razne tužbe i rivalstva, no 2017. najavljen je „Arduino Foundation“, koji bi spojio Arduino LLC i Arduino SRL u jednu tvrtku. Do prve polovice 2019. taj plan još nije proveden.

Slika 2. Prvi prototip Arduina

**2. Hardware Arduina**

**2.1 Uvod**

Arduino je otvoreni hardver, što znači da svatko može razviti vlastiti model Arduino pločice, a svi detalji dizajna pojedinih pločica nalaze se na Arduinovoj stranici. Unatoč otvorenosti projekta, programeri su izdali molbu da sam „Arduino“ ostane ekskluzivan za službene pločice.

Većina modela pločica koristi jedan od Atmel 8-bitnih AVR mikrokontrolera (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) sa različitom količinom memorije, ulaza/izlaza i mogućnosti. Kao ulaze i izlaze pločice koriste redove „ženskih“ konektora, a svaki model ima drugačiji broj konektora.

Sve pločice imaju regulator od 5V, 16MHz kristalni oscilator i keramički rezonator. Pločice također imaju mogućnost nadogradnje u obliku modula, odnosno „shieldova“, koji koriste I2C sabirnicu za komunikaciju. Ti shieldovi mogu imati razne upotrebe. Neke od njih su: komplet senzora (vlaga, temperatura, brzina), shield za mjerenje orijentacije i magnetskog polja, bluetooth modul, flash memorija, itd. Shieldovi se čak mogu spajati jedan sa drugim, što stvara mnogo više mogućnosti.

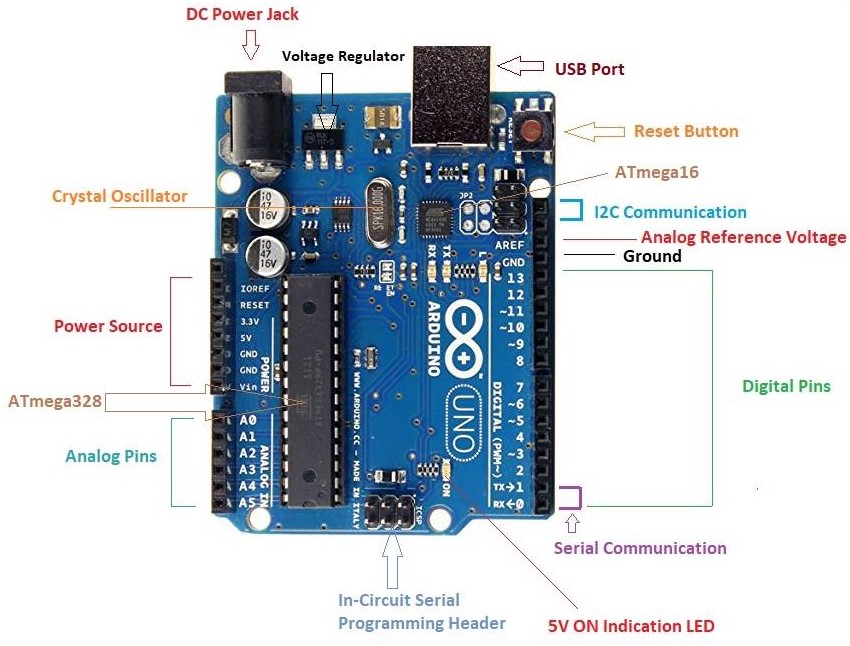
Arduino mikrokontroleri su pre-programirani sa bootloaderom koji pojednostavljuje slanje programa sa računala na flash memoriju pločice. Zadani bootloader svake pločice zove se „Optiboot“. Pločice program dobivaju preko serijske komunikacije sa računalom, odnosno preko USB konekcije.

Do 2019. godine Arduino je izdao 17 verzija pločica, a najpopularnije su:

* Arduino UNO
* Arduino Leonardo
* Arduino Pro
* Arduino Mega
* Arduino Nano
* Arduino LilyPad
* Arduino Robot
* Arduino Ethernet
* Arduino Yun
* Arduino Due

**2.2 Arduino UNO**

Arduino UNO je model pločice baziran na mikrokontroleru ATmega328P. Ima mogućnost napajanja preko USB kabela ili 9V baterije. Sam dizajn se nalazi na službenoj stranici Arduina.

Ime dolazi od talijanske riječi „Uno“, što znači jedan. To ime je odabrano jer je model Arduino UNO bila prva komercijalna pločica. UNO je također prvi u seriji Arduino pločica sa serijskom USB konekcijom.

Slika 3. Raspored Arduino UNO

Na slici su prikazane sve komponente pločice Arduino UNO. Ona sadržava:

* DC ulaz
* Regulator napona – daje 5V na izlazne pinove
* USB priključak – služi za opskrbu napona Arduina, a i preko njega se šalje program na samu pločicu
* Reset tipkalo – u slučaju greške u programu služi za resetiranje Arduina
* LED indikator napajanja – kada je Arduino u pogonu, u stanju HIGH
* 16MHz kristalni oscilator – daje radni takt od 16MHz
* ATmega328P – mikrokontroler pločice, pokreće countere, timere, prekide, radni takt do 20MHz (20 milijuna instrukcija u sekundi)
* 6 Analognih ulaza
* 14 Digitalnih izlaza, 2 za serijsku komunikaciju
* Konektori napajanja (GND, 5V, 3,3V i Vin) – daju napajanje pojedinim komponentama korištenih u projektu

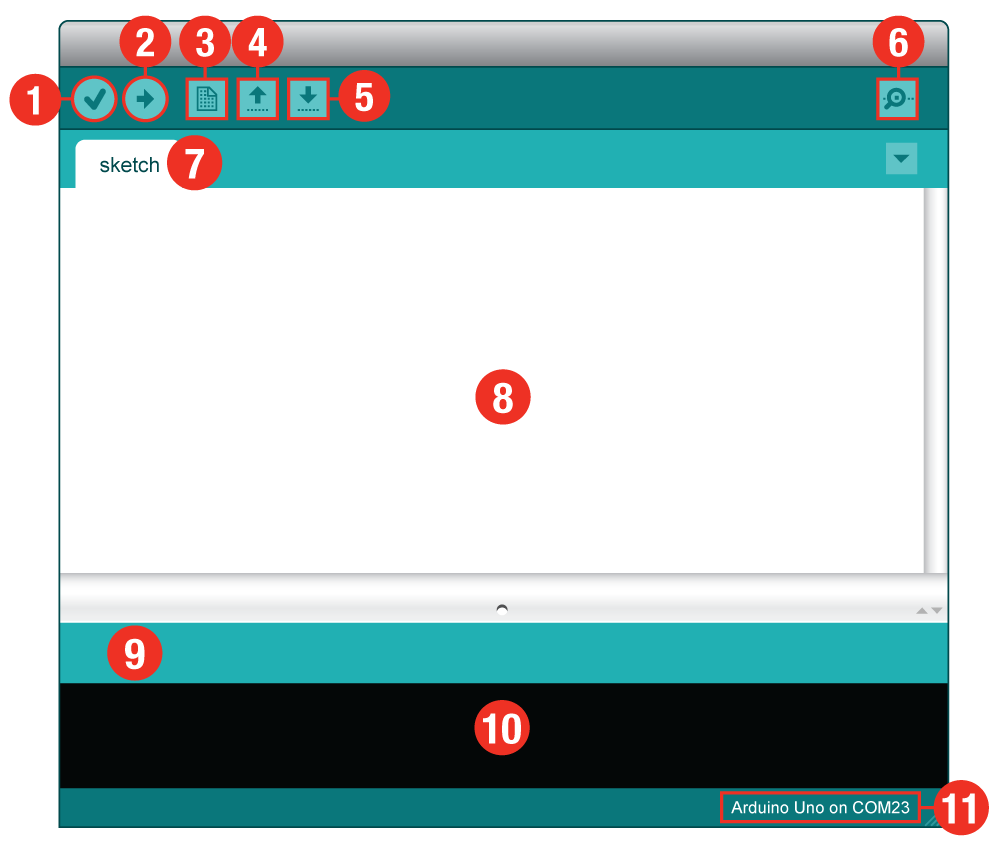
Bitne karakteristike Arduino UNO:

* ATmega328P mikrokontroler
* 5V radni napon
* 20mA za I/O pinove
* 32KB flash memorije
* 2KB SRAM
* 1KB EEPROM
* 16MHz radni takt

**3. Sofware**

**3.1 Arduino IDE**

Arduino integrated development environment (Arduino IDE) je cross-platform aplikacija napisana programskim kodom Java. Služi za pisanje i slanje programskog koda na Arduino pločice, a i na druge tipove pločica (sa dodatnim jezgrama).

Source kod programa je javan na stranici Arduina, i svatko ju može modificirati po želji. Program podržava programski kod C i C++, ali sa vlastitim pravilima strukture. Program napisan u Arduino IDE se zove Sketch. Arduino IDE koristi biblioteke iz „Wiring“ projekta, koji sadrži često korištene funkcije za programiranje mikrokontrolera. Pri prijenosu koda na pločicu, IDE pretvara tekst koda u heksadecimalnu šifru, koja se šalje u memoriju pločice.

Slika 4. Funkcije Arduino IDE

1. Verify: provjerava kod i javlja eventualne greške
2. Upload: šalje kod na Arduino pločicu
3. New: otvaranje novog programa
4. Open: otvaranje postojećeg, spremljenog programa
5. Save: spremanje programa
6. Serial Monitor: otvaranje prozora koji ispisuje komunikaciju sa Arduino pločicom
7. Ime trenutnog sketcha
8. Prostor za pisanje koda: služi za pisanje programskog koda
9. Prostor za poruke: ovdje program govori ako ima grešaka
10. Tekstualna konzola: u ovom prostoru se ispisuju greške u kodu
11. Board i Serial Port: pokazuje informacije o korištenoj pločici i na kojem portu se nalazi

**3.2 Struktura Sketcha**

Sketch je ime za program napravljen u Arduino IDE. Spremljeni su na računalu u tekstualnom obliku sa ekstenzijom *.ino*. Prije verzije 1.0, sketchevi su koristili ekstenziju *.pde*. Svaki sketch se sastoji od dvije glavne funkcije: Setup() i Loop().

Funkcija Setup() se poziva kada se sketch pokreće prvi put. U njoj se postavljaju varijable, OUTPUT i INPUT pinovi, pokretanje Serial Monitora, itd. Funkcija Setup() se odrađuje samo jednom: kada se program pokrene ili resetira. Funkcija Loop() se uvijek poziva nakon funkcije Setup(). Kao što samo ime govori, Loop() se stalno izvršava, odnosno kod u njoj se pokreće dok se program ne ugasi ili ne resetira. Funkcija Loop() se može odgoditi na željeno vrijeme korištenjem funkcije Delay.

Osim funkcija Setup() i Loop(), uglavnom svi programi Arduino IDE sadrže neke osnovne funkcije. Te funkcije su: pinMode, digitalWrite, delay i Serial.begin.

PinMode je najosnovnija funkcija Arduino IDE-a. Pomoću nje definiramo je li određeni pin OUTPUT ili INPUT. Sintaksa: *pinMode(pin, mode).* U sintaksi određujemo je li neki određeni pin INPUT (Arduino prima podatke) ili OUTPUT (Arduino na njega šalje podatke).

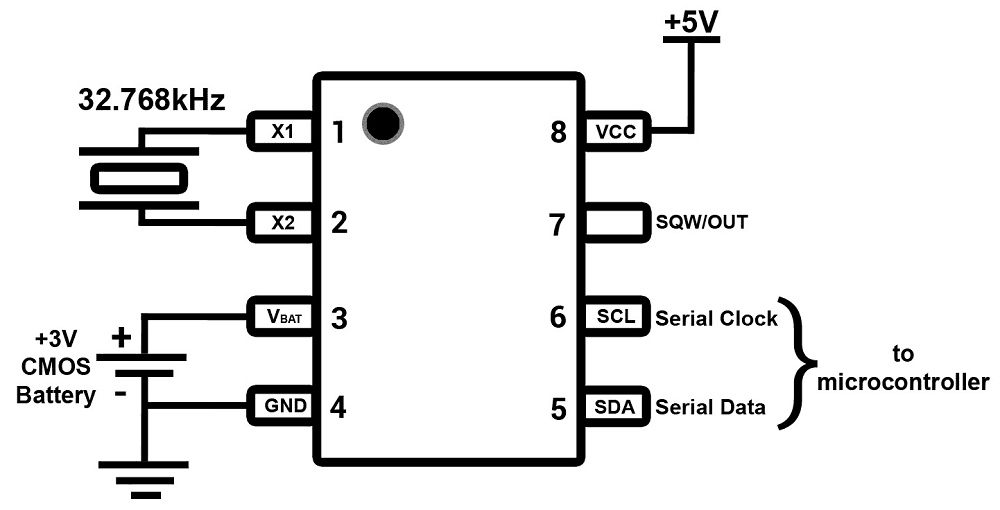
DigitalWrite je funkcija pomoću koje određujemo je li stanje pina HIGH ili LOW. HIGH označuje 5V na određeni pin, a LOW označuje 0V.

Delay se koristi kada želimo u funkciju Loop() dodati kašnjenje. Kašnjenje je definirano u milisekundama, i u to vrijeme Arduino ne obavlja nikakve funkcije.

**4. Izradba digitalnog sata ispisanog riječima**

**4.1 Komponente**

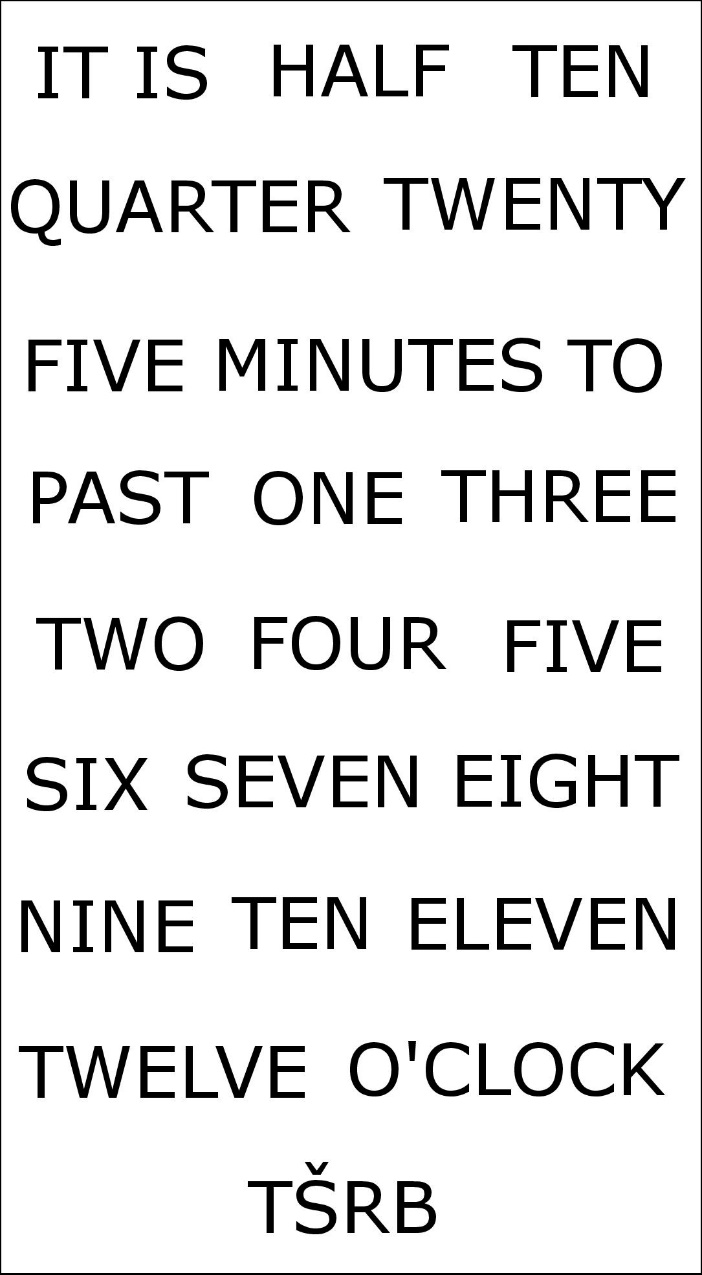
Pri izboru komponenata trebalo je uzeti par stvari u obzir: Real Time Clock (RTC) modul, matricu riječi, kućište i model Arduina.

RTC modul korišten u radu je DS1307+. On je binarni RTC male snage koji ima mogućnost točnog praćenja vremena, datuma, godine i dana u tjednu. Za njegov rad potrebno je imati kvarcni kristal, 3V CMOS baterija i 5V napona. Kvarcni kristal je u iznosu od 32.768kHz, što daje točno praćenje vremena bez dodatnog namještanja. Mogućnost CMOS baterije daje mogućnost praćenja vremena čak i kad je Arduino ugašen, što je korisno u slučaju kvara ili punjenja baterija Arduina.

Slika 5. Shema DS 1307+

Matrica riječi treba sadržavati sve kombinacije kako bismo točno mogli očitati vrijeme, odnosno sate i minute. Vrijeme je izraženo u formatu: IT IS <minuta> PAST/TO <sat> O’CLOCK. Riječi IT IS i O'CLOCK su uvijek upaljene, PAST i TO se izmjenjuju ovisno je li prva ili druga polovica sata. Za varijablu *sat* koriste se brojke od jedan do dvanaest. Za varijablu *minuta* se koriste pet, deset, četvrt, dvadeset i pola.

Ovakav format je izabran jer je jednostavan, ali točan. Da je za sate korišteno od jedan do dvadeset četiri, a za minute od jedan do šezdeset, matrica bi bila, u smislu dimenzija, prevelika, što znači da bi rad u cijelosti bio mnogo kompleksniji, i koristio bi puno više komponenata nego u odabranom formatu.

Riječi su osvjetljene pomoću komadića LED trake. Svaka riječ sadrži jedan komadić LED trake, što znači da ukupno ima 23 komada trake. Korištena LED traka ima tri LED diode po komadiću, a napajanje potrebno za njih je 12V. Napajanje je izvedeno pomoću četiri AA baterije spojene serijski.

Slika 6. Matrica riječi

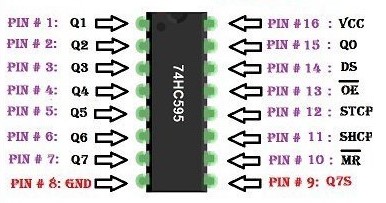
Kućište je plastična kutija dimenzije matrica riječi. Bitni aspekti kućišta su da matrica mora pravilno „sjesti“ u njega i da u njega mora stati Arduino sa pločicom koja sadrži komponente. Visina kućišta je 340mm, širina 195mm, a dubina 160mm. Dubina od 160mm je dovoljna za sve komponente rada. Led diode su smještene na plastičnoj ploči dimenzija matrice udaljena 2cm od same matrice. Kako upaljene diode ne bi osvjetljavale riječi pored sebe, odvojene su plastičnim pregradama.

Na slici 7. se vide same LED diode sa plastičnom pločom i pregradama.

Slika 7. Kućište sa LED diodama

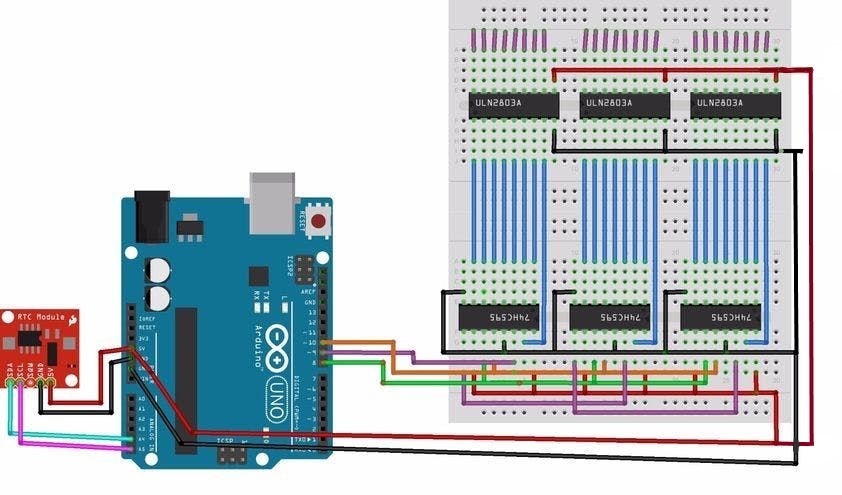
Model Arduina korištenog u radu je Arduino UNO od tvrtke Freenove. UNO je najpoznatiji, najčešće korišten i najjednostavniji model Arduina, što znači da ima najbolju podršku i najviše vodiča za pomoć oko eventualnih greški. Također, Arduino UNO ima najveći izbor kompleta sa raznim dijelovima, kao što su jumperi, otpornici, kondenzatori, senzori, itd.

Problem oko tog modela je što ima samo 14 digitalnih izlaza, dok je za matricu riječi potrebno 23. Najefikasnije rješenje za taj problem su shift registri. Shift registri su sekvencijski logički sklopovi koji pohranjuju i prenose podatke. Oni imaju sposobnost pomicanja („shift“) podataka na sljedeći registar u nizu. Sastoje se od određenog broja „latch“ izlaza, odnosno izlaza koji mogu biti u stanju „0“ ili „1“.

U radu su korištena tri 7HC595 shift registara. Jedan shift registar koristi tri pina na Arduinu, a zauzvrat daje 8 I/O pinova. Kako za matricu treba 23 pinova, tri shift registra daje 24 pinova.

Slika 8. 74HC595 Shift registar

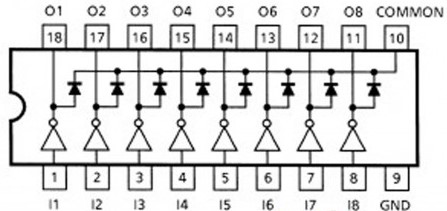
**4.2 Elektronički spoj**

Elektroničke komponente u radu su: Arduino UNO, DS1307+, tri 74HC595 shift registra i tri ULN2803A tranzistorska reda.

Slika 9. Električna shema sata

Na slici 9 može se vidjeti kompletan spoj sata. DS1307+ je spojen na 5V, GND, a njegovi analogni izlazi (pin 5 i pin 6) su spojeni na analogne ulaze A4 i A5 Arduina. Na pinove 1 i 2 spojen je kvarc kristal, a pinovi 3 i 4 sadrže CMOS bateriju.

Shift registri koriste tri pina na Arduinu. Pin 8 je spojen na SRCLK (Serial Clock), preko kojeg registri dobivaju radni takt. Pin 9 spojen je na SER (Serial), na taj pin se serijski šalju bitovi sa Arduina. Pin 10 spojen je na RCLK (Latch), on određuje izlazne vrijednosti shift registara. Na registrima, pinovi 8 (GND) i 13 (Output Enable, negativna logika) su spojeni na masu Arduina. Pinovi 16 (Vcc) i SRCLR (Serial Clear, negativna logika) su spojeni na 5V Arduina. Pin 9 shift registra označuje serijski izlaz. Kada je potrebno rukovati sa više od 8 bitova, potrebno je spojiti serijski izlaz na serijski ulaz sljedećeg registra u nizu, odnosno serijski ulaz prvog registra dobiva podatke sa Arduina, a drugi registar prima podatke sa serijskog izlaza prvog, i tako za sljedeće registre u nizu.

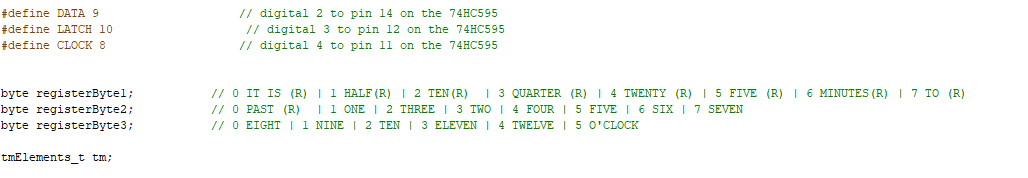
Za pokretanje LED dioda korištena su tri ULN2803A tranzistorska niza. Takav tranzistorski niz ima 8 ulaza i 8 izlaza. Na ulaze prima izlaze shift registara, odnosno vrijednosti bitova za svaku LED diodu. Na izlaze tranzistorskih nizova spojene su mase pojedinih LED dioda.

Slika 10. ULN2803A tranzistorski niz

Kada je na ulazu tranzistora stanje „1“, taj tranzistor se „otvara“, i zatvara strujni krug, odnosno LED diode. Pin 9 (GND) spojen je na masu Arduina, a pin 10 (COMMON) spojen je na 5V Arduina.

**4.3 Objašnjen programski kod**

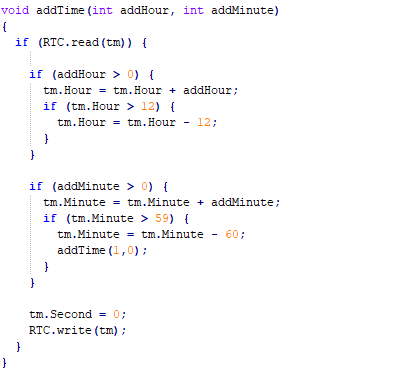
Programski kod sata sadrži dva glavna dijela: čitanje vremena sa RTC-a i pretvaranje tog vremena u bitove koji se šalju na shift registre. Program koristi tri biblioteke: Wire.h sa osnovnim funkcijama registara, Time.h koji sadrži funkcije za praćenje vremena, i DS1307RTC.h koja sadrži sve funkcije za RTC modul DS1307+.

Prvi korak je bio definirati data, latch i clock pinove registara i definirati bitove za svaki registar. Sa naredbom #define definiramo tri pina registra, ta tri pina su u funkciji pinMode zadana kao OUTPUT. Uz pomoć byte varijable zadajemo 24 bita koje koriste tri shift registra.

Slika 11. Definiranje pinova i registara

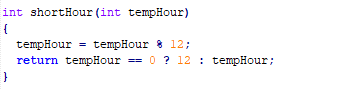
Byte varijabla označuje osam bitova. U komentaru su označena značenja, odnosno LED diode pinova svakoga registra.

tmElements\_t tm označava strukturu koja uvodi u program jednostavan pristup varijablama sati, minuta, datuma i dana tjedna. Ona je potrebna kako bi točno i jednostavno očitali vrijeme sa RTC modula.

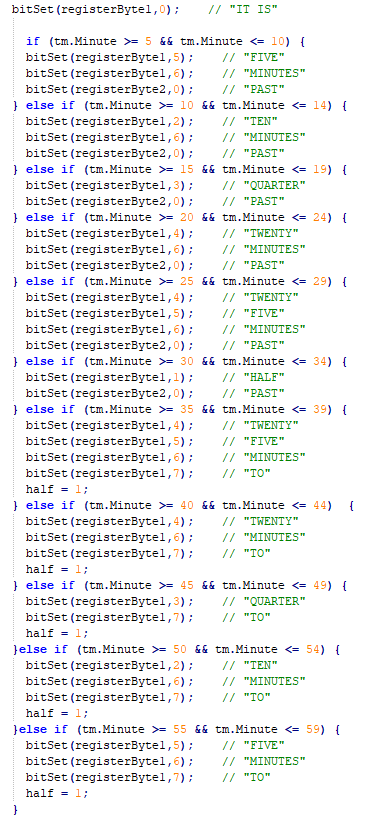
Drugi korak je očitavanje vremena sa RTC modula. To radimo pomoću dvije funkcije iz DS1307.h biblioteke: tm.Hour i tm.Minute. Te dvije funkcije sa RTC modula vrijeme iz binarnog pretvaraju u decimalni oblik. Funkcija addTime očitava vrijeme sa RTC modula koristeći te dvije funkcije.

Slika 12. Definiranje varijabli sati i minuta

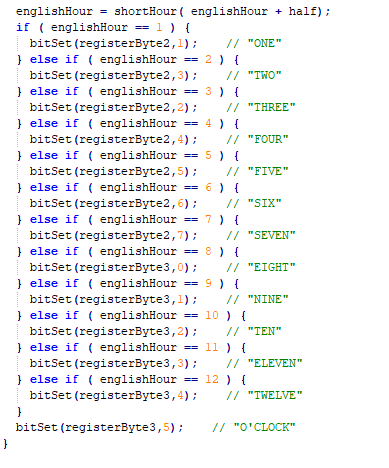
S obzirom da je oblik vremena sa RTC modula 24 satni, zbog 12 satnoga oblika matrice potrebno ga je preoblikovati. To je druga svrha funkcije addTime. Ako vrijednost sati pređe 12, jednostavno se ta vrijednost oduzme sa 12, i rezultat je konačna vrijednost sata. Također se brine za vrijednost minute, ako se dogodi greška i vrijednost minute prijeđe 59, oduzima se sa 60 kako bi se vratio pravilan redoslijed vremena, i vrijednosti sati se dodaje +1. Na kraju svih uvjeta, na RTC šaljemo dobivene podatke funkcijom RTC.write(tm).

Zadnji korak očitavanja vremena je uvjet funkcije shortHour. Taj uvjet govori: ako je trenutna vrijednost sati nula, funkcija vraća dvanaest. Ako nije nula, funkcija vraća vrijednost sati.

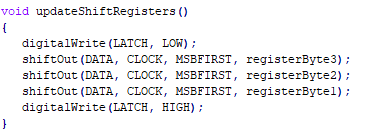
Slika 13. Uvjet shortHour

Treći korak je definiranje uvjeta koji uključuju potrebne LED diode. Ti uvjeti se nalaze u funkciji timeToBytes. timeToBytes se sastoji od dva dijela: minute i sati. Uvjeti za minute se dobivaju pomoću tm.Minute varijable iz RTC modula. Uvjeti provjeravaju u kojem je rasponu trenutna vrijednost minuta, pa ovisno o tome se pale određene LED diode (npr. ako je trenutno u šestoj minuti, pale se diode FIVE MINUTES PAST).

Slika 14. Uvjeti LED dioda za minute

Za sate se koristi varijabla englishHour. englishHour se sastoji od zbroja varijabli shortHour iz prijašnjeg koraka i half. Half varijabla je u nuli prvih trideset minuta, a drugih trideset joj je vrijednost jedan. Njena funkcija je ta da određuje pali li se LED dioda trenutnog sata ili sljedećeg, ovisno o PAST ili TO formatu (npr. vrijeme 3:05 pali FIVE MINUTES PAST THREE, a 3:55 FIVE MINUTES TO FOUR). Bitovi se šalju pomoću funkcije bitSet, u kojoj definiramo koji bit od 24 će biti u stanju „1“.

Slika 15. Uvjeti LED dioda za sate

Na kraju svih uvjeta se poziva funkcija updateShiftRegisters, koja je zapravo tri shiftOut funkcije za svaki shift registar. shiftOut je funkcija specifična za shift registre. Ona šalje nizove bitova na zadane pinove registra. Prije slanja podataka treba namjestiti LATCH pin u stanje LOW. Šaljemo podatke na DATA i CLOCK. MSBFIRST u funkciji označava redoslijed bitova koji se šalju na registre. MSBFIRST označava „Most Significant Bit First“.

Slika 16. Funkcija updateShiftRegisters

**5. Programski kod sata**

#include <Wire.h>

#include <Time.h>

#include <DS1307RTC.h>

#define DATA 9 // digital 2 to pin 14 on the 74HC595

#define LATCH 10 // digital 3 to pin 12 on the 74HC595

#define CLOCK 8 // digital 4 to pin 11 on the 74HC595

byte registerByte1**;** // 0 IT IS (R) | 1 HALF(R) | 2 TEN(R) | 3 QUARTER (R) | 4 TWENTY (R) | 5 FIVE (R) | 6 MINUTES(R) | 7 TO (R)

byte registerByte2**;** // 0 PAST (R) | 1 ONE | 2 THREE | 3 TWO | 4 FOUR | 5 FIVE | 6 SIX | 7 SEVEN

byte registerByte3**;** // 0 EIGHT | 1 NINE | 2 TEN | 3 ELEVEN | 4 TWELVE | 5 O'CLOCK

tmElements\_t tm**;**

void setup**()**

**{**

Serial**.**begin**(**9600**);**

**while** **(!**Serial**)** **;**

delay**(**200**);**

Serial**.**println**(**"DS1307RTC Read Test"**);**

Serial**.**println**(**"-------------------"**);**

pinMode**(**LATCH**,** OUTPUT**);**

pinMode**(**CLOCK**,** OUTPUT**);**

pinMode**(**DATA**,** OUTPUT**);**

**}**

void loop**()**

**{**

timeToBytes**();**

updateShiftRegisters**();**

printTime**();**

**}**

void timeToBytes**()**

**{**

**if** **(**RTC**.**read**(**tm**))** **{**

int englishHour **=** shortHour**(**tm**.**Hour**);**

int half **=** 0**;**

registerByte1 **=** 0**;**

registerByte2 **=** 0**;**

registerByte3 **=** 0**;**

bitSet**(**registerByte1**,**0**);** // "IT IS"

**if** **(**tm**.**Minute **>=** 5 **&&** tm**.**Minute **<=** 10**)** **{**

bitSet**(**registerByte1**,**5**);** // "FIVE"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 10 **&&** tm**.**Minute **<=** 14**)** **{**

bitSet**(**registerByte1**,**2**);** // "TEN"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 15 **&&** tm**.**Minute **<=** 19**)** **{**

bitSet**(**registerByte1**,**3**);** // "QUARTER"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 20 **&&** tm**.**Minute **<=** 24**)** **{**

bitSet**(**registerByte1**,**4**);** // "TWENTY"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 25 **&&** tm**.**Minute **<=** 29**)** **{**

bitSet**(**registerByte1**,**4**);** // "TWENTY"

bitSet**(**registerByte1**,**5**);** // "FIVE"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 30 **&&** tm**.**Minute **<=** 34**)** **{**

bitSet**(**registerByte1**,**1**);** // "HALF"

bitSet**(**registerByte2**,**0**);** // "PAST"

**}** **else** **if** **(**tm**.**Minute **>=** 35 **&&** tm**.**Minute **<=** 39**)** **{**

bitSet**(**registerByte1**,**4**);** // "TWENTY"

bitSet**(**registerByte1**,**5**);** // "FIVE"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte1**,**7**);** // "TO"

half **=** 1**;**

**}** **else** **if** **(**tm**.**Minute **>=** 40 **&&** tm**.**Minute **<=** 44**)** **{**

bitSet**(**registerByte1**,**4**);** // "TWENTY"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte1**,**7**);** // "TO"

half **=** 1**;**

**}** **else** **if** **(**tm**.**Minute **>=** 45 **&&** tm**.**Minute **<=** 49**)** **{**

bitSet**(**registerByte1**,**3**);** // "QUARTER"

bitSet**(**registerByte1**,**7**);** // "TO"

half **=** 1**;**

**}else** **if** **(**tm**.**Minute **>=** 50 **&&** tm**.**Minute **<=** 54**)** **{**

bitSet**(**registerByte1**,**2**);** // "TEN"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte1**,**7**);** // "TO"

half **=** 1**;**

**}else** **if** **(**tm**.**Minute **>=** 55 **&&** tm**.**Minute **<=** 59**)** **{**

bitSet**(**registerByte1**,**5**);** // "FIVE"

bitSet**(**registerByte1**,**6**);** // "MINUTES"

bitSet**(**registerByte1**,**7**);** // "TO"

half **=** 1**;**

**}**

englishHour **=** shortHour**(** englishHour **+** half**);**

**if** **(** englishHour **==** 1 **)** **{**

bitSet**(**registerByte2**,**1**);** // "ONE"

**}** **else** **if** **(** englishHour **==** 2 **)** **{**

bitSet**(**registerByte2**,**3**);** // "TWO"

**}** **else** **if** **(** englishHour **==** 3 **)** **{**

bitSet**(**registerByte2**,**2**);** // "THREE"

**}** **else** **if** **(** englishHour **==** 4 **)** **{**

bitSet**(**registerByte2**,**4**);** // "FOUR"

**}** **else** **if** **(** englishHour **==** 5 **)** **{**

bitSet**(**registerByte2**,**5**);** // "FIVE"

**}** **else** **if** **(** englishHour **==** 6 **)** **{**

bitSet**(**registerByte2**,**6**);** // "SIX"

**}** **else** **if** **(** englishHour **==** 7 **)** **{**

bitSet**(**registerByte2**,**7**);** // "SEVEN"

**}** **else** **if** **(** englishHour **==** 8 **)** **{**

bitSet**(**registerByte3**,**0**);** // "EIGHT"

**}** **else** **if** **(** englishHour **==** 9 **)** **{**

bitSet**(**registerByte3**,**1**);** // "NINE"

**}** **else** **if** **(** englishHour **==** 10 **)** **{**

bitSet**(**registerByte3**,**2**);** // "TEN"

**}** **else** **if** **(** englishHour **==** 11 **)** **{**

bitSet**(**registerByte3**,**3**);** // "ELEVEN"

**}** **else** **if** **(** englishHour **==** 12 **)** **{**

bitSet**(**registerByte3**,**4**);** // "TWELVE"

**}**

bitSet**(**registerByte3**,**5**);** // "O'CLOCK"

**}**

**}**

void updateShiftRegisters**()**

**{**

digitalWrite**(**LATCH**,** LOW**);**

shiftOut**(**DATA**,** CLOCK**,** MSBFIRST**,** registerByte3**);**

shiftOut**(**DATA**,** CLOCK**,** MSBFIRST**,** registerByte2**);**

shiftOut**(**DATA**,** CLOCK**,** MSBFIRST**,** registerByte1**);**

digitalWrite**(**LATCH**,** HIGH**);**

**}**

void addTime**(**int addHour**,** int addMinute**)**

**{**

**if** **(**RTC**.**read**(**tm**))** **{**

**if** **(**addHour **>** 0**)** **{**

tm**.**Hour **=** tm**.**Hour **+** addHour**;**

**if** **(**tm**.**Hour **>** 12**)** **{**

tm**.**Hour **=** tm**.**Hour **-** 12**;**

**}**

**}**

**if** **(**addMinute **>** 0**)** **{**

tm**.**Minute **=** tm**.**Minute **+** addMinute**;**

**if** **(**tm**.**Minute **>** 59**)** **{**

tm**.**Minute **=** tm**.**Minute **-** 60**;**

addTime**(**1**,**0**);**

**}**

**}**

tm**.**Second **=** 0**;**

RTC**.**write**(**tm**);**

**}**

**}**

int shortHour**(**int tempHour**)**

**{**

tempHour **=** tempHour **%** 12**;**

**return** tempHour **==** 0 **?** 12 **:** tempHour**;**

**}**

void printTime**()** **{**

**if** **(**RTC**.**read**(**tm**))** **{**

Serial**.**print**(**"Ok, Time = "**);**

print2digits**(**tm**.**Hour**);**

Serial**.**write**(**':'**);**

print2digits**(**tm**.**Minute**);**

Serial**.**write**(**':'**);**

print2digits**(**tm**.**Second**);**

Serial**.**print**(**", Date (D/M/Y) = "**);**

Serial**.**print**(**tm**.**Day**);**

Serial**.**write**(**'/'**);**

Serial**.**print**(**tm**.**Month**);**

Serial**.**write**(**'/'**);**

Serial**.**print**(**tmYearToCalendar**(**tm**.**Year**));**

Serial**.**println**();**

Serial**.**print**(**"registerByte3 = "**);**

Serial**.**print**(**registerByte3**,** BIN**);**

Serial**.**println**();**

Serial**.**print**(**"registerByte2 = "**);**

Serial**.**print**(**registerByte2**,** BIN**);**

Serial**.**println**();**

Serial**.**print**(**"registerByte1 = "**);**

Serial**.**print**(**registerByte1**,** BIN**);**

Serial**.**println**();**

**}** **else** **{**

**if** **(**RTC**.**chipPresent**())** **{**

Serial**.**println**(**"The DS1307 is stopped. Please run the SetTime"**);**

Serial**.**println**(**"example to initialize the time and begin running."**);**

Serial**.**println**();**

**}** **else** **{**

Serial**.**println**(**"DS1307 read error! Please check the circuitry."**);**

Serial**.**println**();**

**}**

delay**(**9000**);**

**}**

delay**(**1000**);**

**}**

void print2digits**(**int number**)** **{**

**if** **(**number **>=** 0 **&&** number **<** 10**)** **{**

Serial**.**write**(**'0'**);**

**}**

Serial**.**print**(**number**);**

**}**

**6. Zaključak**

Izabrao sam ovaj završni rad jer je zanimljiv i kreativan način vrlo jednostavne radnje: očitavanje vremena. Puno je zanimljivije pročitati „IT IS FIFTEEN MINUTES PAST THREE“ nego pogledati na analogni sat ili na mobitel. Uz to, takvi satovi su vrlo rijetki, a gotovi proizvodi nose visoku cijenu. Također sam naučio puno o shift registrima i RTC modulima, a i o njihovim glavnim funkcijama. Kako je ovaj projekt baziran na programskom kodu, program je vjerojatno mogao biti kraći i efikasniji, pa se mogu napraviti poboljšanja.

**7. Literatura**

<https://en.wikipedia.org/wiki/Arduino>

<https://en.wikipedia.org/wiki/Arduino_Uno>

<https://en.wikipedia.org/wiki/Shift_register>

<https://forum.arduino.cc/>

<https://www.arduino.cc/en/Guide/HomePage>

<https://www.arduino.cc/en/guide/environment>

<https://www.arduino.cc/en/Reference/FunctionDeclaration>

<https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/understanding-the-code>

<https://www.instructables.com/id/How-Shift-Registers-Work-74HC595/>

<https://forum.allaboutcircuits.com/threads/uln2803-how-does-it-work.32267/>