TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA

GETALDIĆEVA 4, ZAGREB

**Završni stručni rad:**

**Sat s Lixie segmentima**

Mentor: Učenik:

Zoran Dumančić, mag.ing.el. Mate Renić, 4.H

Zagreb, travanj 2020.

**Projektni zadatak**

Teze projektnog zadatka:

Konstruirati sat od 6 Lixie segmenata koji će prikazivati sate, minute i sekunde na Arduino platformi. Ugraditi izbornik koji će korisniku omogućavati podešavanje odbrojavanja (counter), podešavanje alarma (sa zvukom) te štoperica. Omogućiti i izbor boje. Izbornikom upravljati pomoću inkrementalnog enkodera ili joystickom. Omogućiti podešavanje svjetline ručno ili u skladu s vanjskim osvjetljenjem. Konačna verzija rada treba biti ugrađena u odgovarajuće kućište.

Sadržaj

[**1. Uvod** 2](#_Toc41326448)

[**2. Opis tehnologija i korištenih programa** 3](#_Toc41326449)

[**2.1 Lixie tehnologija** 3](#_Toc41326450)

[**2.2 Arduino Uno** 4](#_Toc41326451)

[**2.3 Arduino IDE** 5](#_Toc41326452)

[**2.4 I2C serijska komunikacija** 6](#_Toc41326453)

[**3. Opis rada** 8](#_Toc41326454)

[**3.1 Korištene komponente** 9](#_Toc41326455)

[**3.1.1 Lixie segmenti** 9](#_Toc41326456)

[**3.1.2 RGB LED traka WS2812B** 10](#_Toc41326457)

[**3.1.3 Sat realnog vremena** 12](#_Toc41326458)

[**3.1.4 Okretni davač** 13](#_Toc41326459)

[**3.1.5 LCD zaslon** 14](#_Toc41326460)

[**3.1.6 Piezo zujalica** 15](#_Toc41326461)

[**3.2 Izrada rada** 16](#_Toc41326462)

[**3.3 Konačni izgled rada** 19](#_Toc41326463)

[**4. Bitni dijelovi programa** 21](#_Toc41326464)

[**4.1 Korištene biblioteke** 21](#_Toc41326465)

[**4.2 Učitavanje korisničkih postavki** 21](#_Toc41326466)

[**4.3 Inicijalne postavke programa** 22](#_Toc41326467)

[**4.4 Čitanje vremena s RTC modula** 22](#_Toc41326468)

[**4.5 Određivanje brojeva koji će se prikazati na Lixie segmentima** 23](#_Toc41326469)

[**4.6 Ispisivanje vremena na Lixie segmentima** 25](#_Toc41326470)

[**4.7 Ulazak u izbornik** 26](#_Toc41326471)

[**4.8 Alarmi** 26](#_Toc41326472)

[**4.9 Izbornik** 27](#_Toc41326473)

[**4.10 Prekid za okretni davač** 29](#_Toc41326474)

[**5. Zaključak** 30](#_Toc41326475)

[**6. Literatura** 31](#_Toc41326476)

[**7. Prilozi** 32](#_Toc41326477)

[**7.1 Cjelokupni programski kod** 32](#_Toc41326478)

# **1. Uvod**

U ljeto 2018. godine gledao sam jednu seriju. Cijeli koncept je bio baziran na putovanju kroz vrijeme, no to nije toliko bitno. Važno je ono što sam vidio u toj seriji. Bio je to sat napravljeno od starih Nixie cijevi. To su segmenti koji sadrže nekoliko vodiča u sebi te ovisno kroz koji prolazi električna energije, svijetli taj vodič. Ti vodiči su oblikovani kao brojevi. Tada sam prvi put ih ugledao. I svidjelo mi se kako izgleda. Bili su ugodne narančaste boje. I tako je započelo moje istraživanje. Znao sam da ću u 4. razredu Tehničke škole Ruđera Boškovića morati napraviti završni rad pa sam mislio koristiti Nixie cijevi. No zbog velike cijene sam bio razočaran. Ali među poveznicama na razne stranice nalazio se jedan poseban link koji je vodio na nešto vrlo slično. Bili su to Lixie segmenti. Iako nisu imali isti dizajn niti bili uopće slični Nixie cijevima, vidio sam jednu mogućnost. Zanimljiv izgled me zainteresirao i obećao sam si da ću napraviti sat od Lixie segmenata. Iako se ispočetka činio vrlo jednostavno, bilo je ponekih prepreka u izradi. I evo me ovdje, na početku dokumentacije kroz koji ću opisati sve što sam napravio kako bih realizirao navedeni projekt te što sam sve naučio tijekom izrade projekta.

# **2. Opis tehnologija i korištenih programa**

## **2.1 Lixie tehnologija**

Ključni princip na kojem radi Lixie tehnologija je lom svjetlosti koja dolazi sa RGB LED dioda WS2812B na akrilnim pločicama koje imaju ugravirane brojeve. Postavljanjem tih akrilnih pločica direktno na diode ili filtriranje svjetlosti kako bi išla na određene brojeve, osigurava se dobra vidljivost brojeva. Svjetlost koja ulazi u akril prolazi cijelom dužinom akrila dok ne dođe do kraja ili ugraviranog područja. Tada se svjetlost rasprši. Upravo zbog tog fenomena je moguće dobiti boju samo na broju i rubovima segmenta, a ne na cijelom području akrilne pločice [1].



Slika 2.1.2 Pogled na Lixie segment sa desne strane

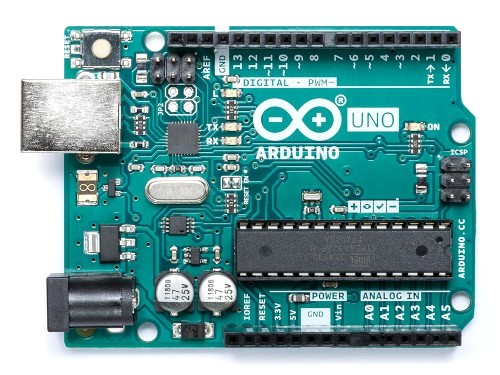


Slika 2.1.1 Pogled na Lixie segment od naprijed

Na slikama 2.1.1 i 2.1.2 se može vidjeti primjer navedene tehnologije. Osvijetljena je samo brojka šest i rubovi akrilne pločice.

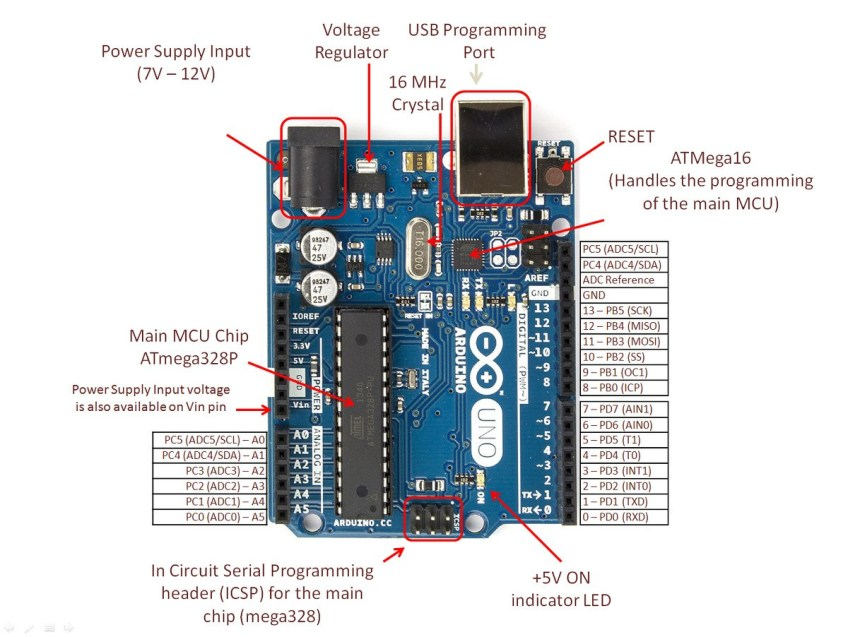
## **2.2 Arduino Uno**

Arduino Uno je baziran na ATmega328p mikroprocesoru napredne RISC arhitekture. Na čipu se nalaze 3 memorije – 32KB Flash memorije za pohranu koda, 2KB SRAM-a za privremeno spremanje podataka tijekom rada te 1KB EEPROM-a za pohranu podataka varijabli u svrhu nekih postavki kako se ne bi izgubilo nakon isključivanja.



Slika 2.2.1 Arduino Uno pločica [2]

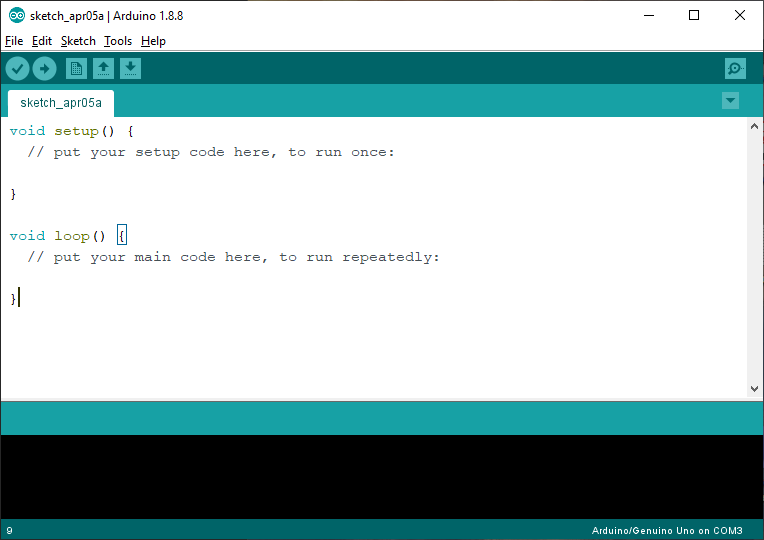
Uz navedene memorije, na čipu nalazimo 2 seta registara – 32 interna registra AVR procesora i 64 ulazno/izlaznih registara + 160 proširenih registara. ATmega328p ima 3 porta – B (8 pinova), C (7 pinova) i D (8 pinova). Neki od pinova nisu izvedeni zbog uštede prostora. Među pinovima razlikujemo digitalne i analogne. Digitalni pinovi daju/primaju 0 ili 1 kao vrijednost dok analogni pinovi daju/primaju vrijednosti od 0 do 255. Također zbog uštede prostora, većina pinova ima 1 ili više funkcija.



Slika 2.2.2 Raspored izvoda na Arduino Uno [3]

## **2.3 Arduino IDE**

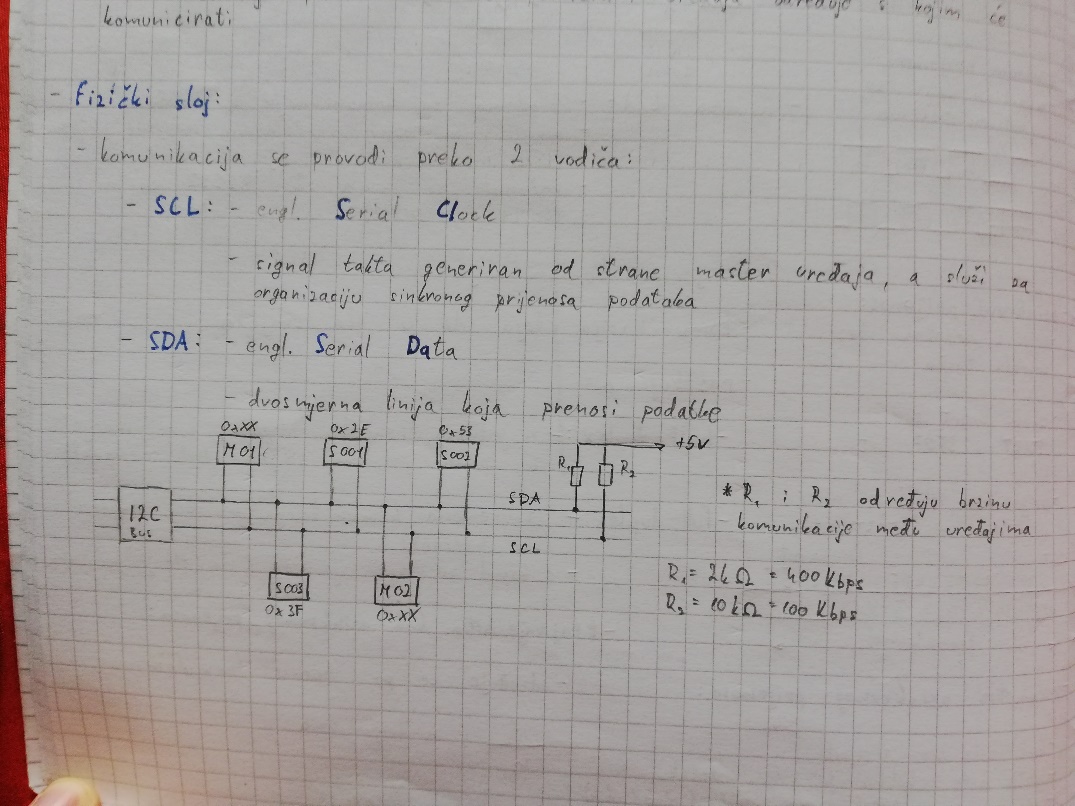
Arduino IDE je program za programiranje mikroupravljača Arduino platforme. Program je jednostavan za korištenje i sadrži sve što bi prosječnom korisniku moglo zatrebati pri izradi programa za Arduino mikroupravljače. Glavni prozor programa sastoji se od alatne trake, prostora za izradu koda i prostor koji daje povratne informacije o mogućim greškama te stanju samog mikroupravljača. Serial monitor je jedan od korisnih alata korišten za dobivanje povratnih informacija sa Arduina. U ovom projektu se koristi samo u „void setup()“ pri inicijalizaciji RTC modula.



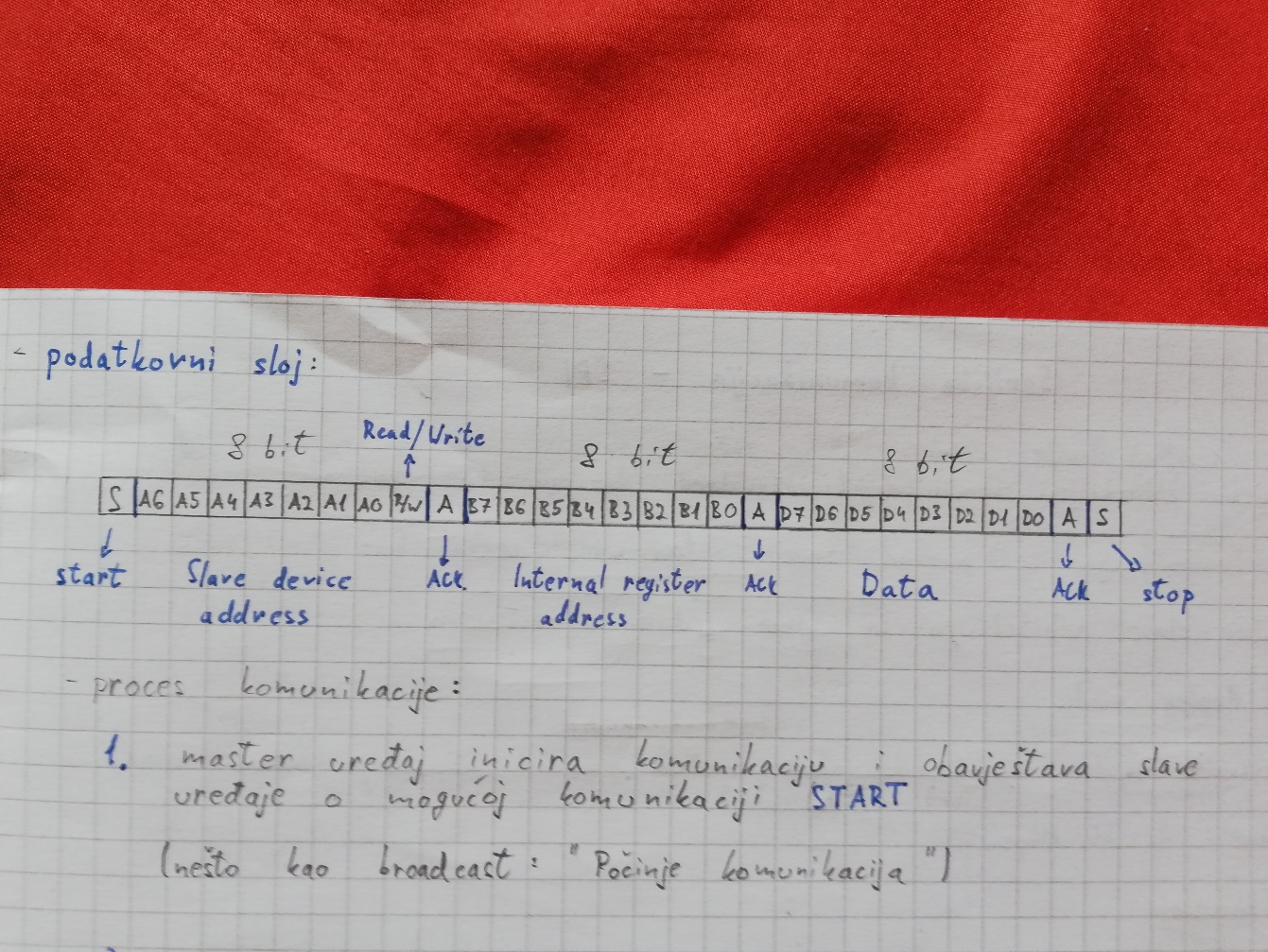
1. Crveni okvir predstavlja alatnu traku
2. Žuti okvir predstavlja prostor za izradu koda
3. Zeleni okvir predstavlja prostor sa povratnim informacijama

## **2.4 I2C serijska komunikacija**

I2C je serijski bidirekcijski komunikacijski standard baziran na sinkronoj komunikaciji. Tvrtka Philip Semiconductor razvila ga je 1982. godine. Implementacija je vrlo jednostavna. Koristi se za povezivanje sporijih perifernih uređaja na CPU preko relativno kratkih udaljenosti. Za ostvarivanje komunikacije su potrebna samo 2 vodiča -SDA i SCL. Komunikacija je 8-bitna. Koncept komunikacije baziran je na Master-Slave odnosu. Treba biti minimalno jedan od obje vrste uređaja. Svaki uređaj ima predefinirani jedinstveni identifikator (ID). Master uređaj prema jedinstvenom indikatoru slave uređaja određuje s kojim će komunicirati. Ovaj način komunikacije se koristi za komunikaciju sa LCD zaslonom [4].



Slika 2.4.1 Objašnjenje spajanja i vodiča SDA i SCL



Slika 2.4.2 Oblik paketa I2C komunikacije

Koraci u komunikaciji:

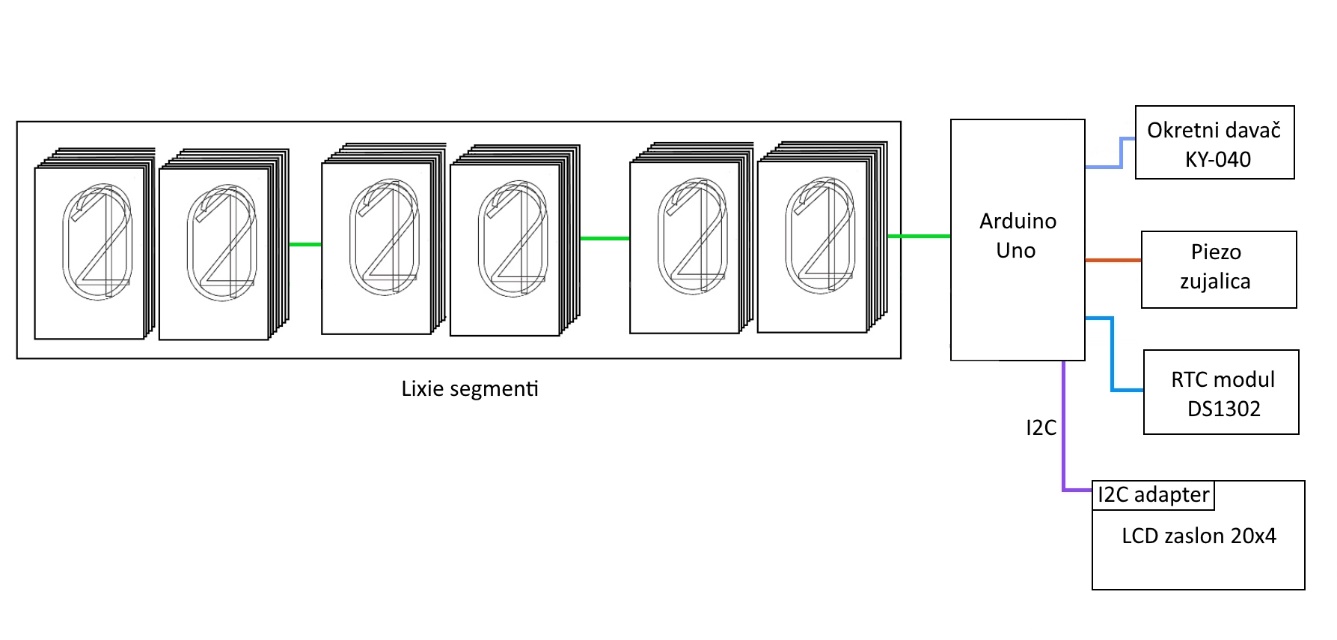
1. Master uređaj inicira komunikaciju i obavještava slave uređaj o mogućoj komunikaciji START (nešto kao broadcast: „Počinje komunikacija“)
2. Master uređaj na SDA liniju postavlja adresu slave uređaja s kojim želi ostvariti komunikaciju i želi li dobiti podatke ili ih zapisati kod slave uređaja (ostali slave-ovi u „Z“ stanju (kao da nije spojen))
3. Slave uređaj odgovara sa ACK signalom
4. Nakon ACK signala, master uređaj šalje adresu internog registra slave uređaja koji želi čitati ili izmjenjivati
5. Slave uređaj odgovara sa ACK signalom
6. Master uređaj šalje podatke ili čita podatke određenog slave uređaja
7. Slave uređaj odgovara sa ACK signalom
8. Mater uređaj završava komunikaciju sa STOP

# **3. Opis rada**

Tema projekta je „Sat s Lixie segmentima“. Ovaj rad je odabran zbog zanimljivog izgleda, želje za izazovom i potrebom za satom u sobi. Cilj projekta je ostvariti sat baziran na Arduino platformi koji će prikazivati sate, minute i sekunde. Segmenti koji će prikazivati vrijeme sastavljeni su od malih akrilnih pločica sa ugraviranim brojevima. Te brojke će osvjetljavati RGB LED diode na dnu. Vrijeme se dobiva i prati sa RTC modulom. Uz funkcionalnosti sata dodaje se izbornik koji će korisniku omogućiti izbor boje i svjetline RGB LED dioda, namještanje alarma i zvuka za alarm te će imati i ugrađenu opciju za štopericu. Izbornikom se upravlja putem okretnog davača. Okretanjem u desno, izbornik ide prema dolje do ostalih opcija dok ne dođe do zadnje opcije. Okretanjem u lijevo, izbornik ide prema gore dok ne dođe do prve opcije. Izbornik je na engleskom jeziku. Struktura izbornika je iduća:

1. Indikator koji ukazuje da se korisnik nalazi u izborniku
2. Tri opcije od kojih korisnik može birati

Slika 3.1 Blok shema spoja



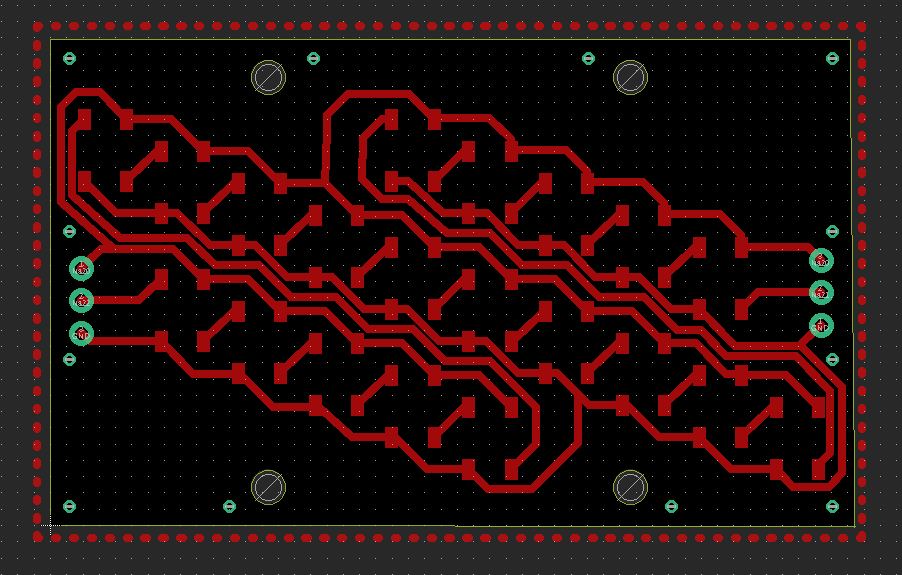
Opcije izbornika su:

* Mijenjanje svjetline Lixie segmenata
* Mijenjanje boje Lixie segmenata
* Biranje tona alarma
* Postavljanje i brisanje vremena alarma
* Korištenje štoperice
* Izlaz iz izbornika

## **3.1 Korištene komponente**

### **3.1.1 Lixie segmenti**

Lixie segmenti su glavni dio ovog projekta. Oni služe za ispis vremena. Sastoje se od više pojedinačnih komponenti. Kao bazu imaju tiskane pločice sa tri vodiča – napajanje (+5V), podaci (Data) i uzemljenje (GND). Dimenzije tiskanih pločica su 62.30x38 mm.

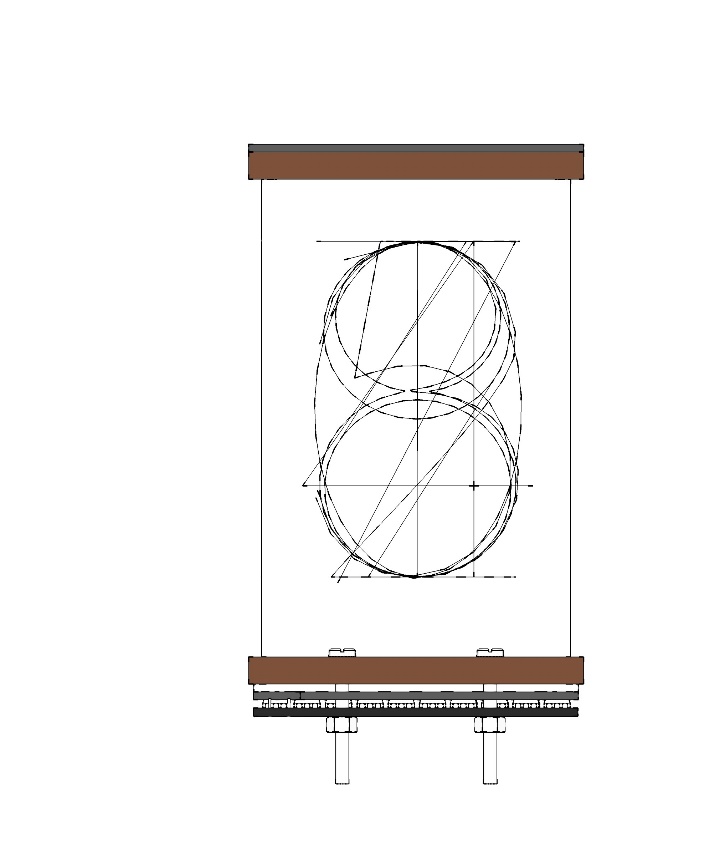


Slika 3.1.1.1 Prikaz vodiča na tiskanoj pločici pomoću programa EAGLE [5 (modificirano)]

Na tiskane pločice zalemljene su RGB LED diode WS2812B koje su opisane u nastavku. Na diode je postavljen blok akrilnih pločica u koje su ugravirani brojevi od 0 do 9. Dimenzije akrilnih pločica su 100x60x2 mm sa dubinom graviranja brojki od 0.5mm [5].



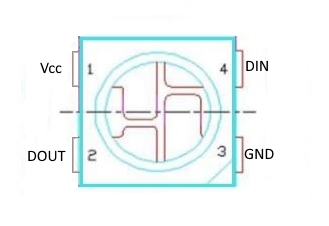
Slika 3.1.1.3 Prikaz originalnog Lixie segmenta sa strane [5]



Slika 3.1.1.2 Prikaz originalnog Lixie segmenta od naprijed [5]

### **3.1.2 RGB LED traka WS2812B**

WS2812B je ime za RGB LED diode sa pametnim kontrolnim i RGB čipom koji su integrirani u kućište veličine 5x5 mm. Svim osnovnima bojama može se mijenjati jačina svjetlosti na skali od 0 (ugašena) do 256 (maksimum). Sveukupno može pokriti 16777216 (256^3) kombinacija boja. Više ovakvih segmenata spaja se paralelno za napajanje i uzemljenje dok se podaci prenose jednom linijom kroz sve segmente.



Slika 3.1.2.1 Raspored pinova RGB LED diode WS2812B [6]

Na jednom segmentu nalaze se četiri kontakta. Jedan za napajanje, jedan za uzemljenje, jedan za primanje signala i jedan za predaju signala dalje. Zbog degradacije signala, svaka komponenta ima pojačivač kako se signal ne bi izgubio. U ovom projektu na jednu tiskanu pločicu ide 20 WS2812B segmenata. One će osvjetljavati akrilne pločice sa ugraviranim brojevima. Boje koje će biti dostupne su crvena, zelena, plava i narančasta. Ove RGB LED diode su SMD tehnologije. To u engleskom jeziku znači „Surface Mounted Diode“. Obično se taj akronim može vidjeti kod LED dioda koje nemaju klasične kontakte koji ulaze u rupe već se stavljaju na površinu i onda leme na vodiče [6].

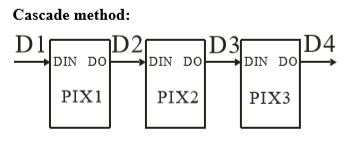


Vcc DIN GND

Vcc DOUT GND

Slika 3.1.2.2 Raspored RGB LED dioda WS2812B na tiskanoj pločici

Spajanje je izvedeno tako da se diode nalaze u kaskadi te izlaz podataka sa jedne ide na ulaz podataka druge diode. Unutar same diode se podatak modificira i pojačava kako bi mogao doći do kraja linije dioda [6].



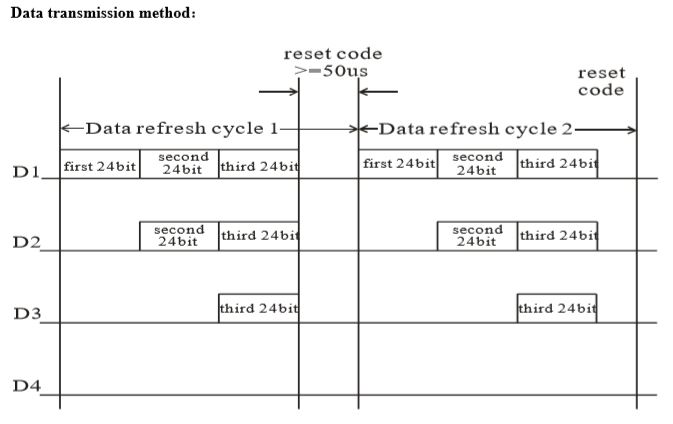
Slika 3.1.2.3 Prikaz kaskadnog načina spajanja dioda WS2812B [6]

Podaci koji se šalju i koriste za upravljanje diodama su veličine 24 bita. Za svaku boju se nalazi 8 dijelova (sa maksimalnom vrijednošću od 255) [6].



Slika 3.1.2.4 Oblik paketa koji diode interpretiraju za boju [6]

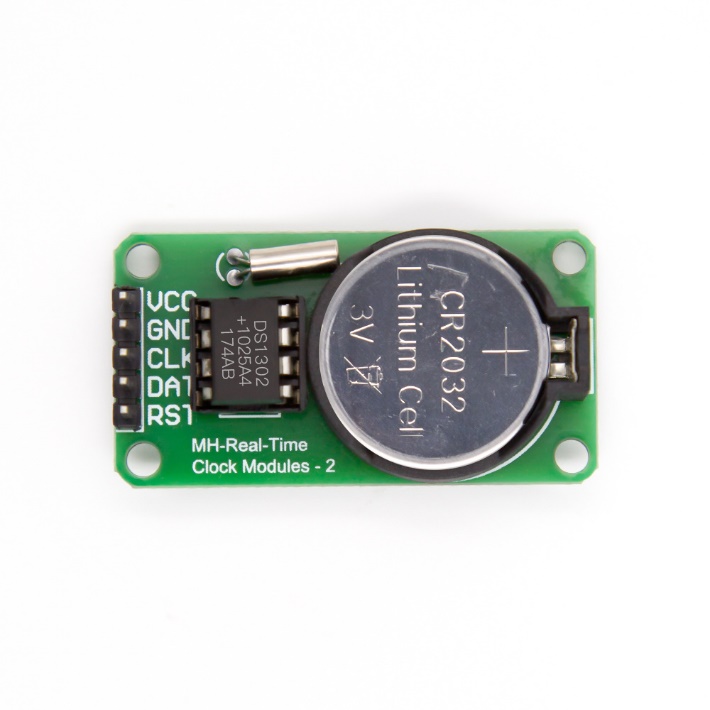
Diode znaju za koju od njih je podatak prema vremenu slanja u jednom ciklusu. Vrijeme jednog ciklusa se mijenja s obzirom na broj dioda u jednom nizu. Duljina Reset signala je uvijek ista [6] .



Slika 3.1.2.5 Prikaz dvaju ciklusa sa četiri diode gdje se četvrta dioda ne koristi [6]

### **3.1.3 Sat realnog vremena**

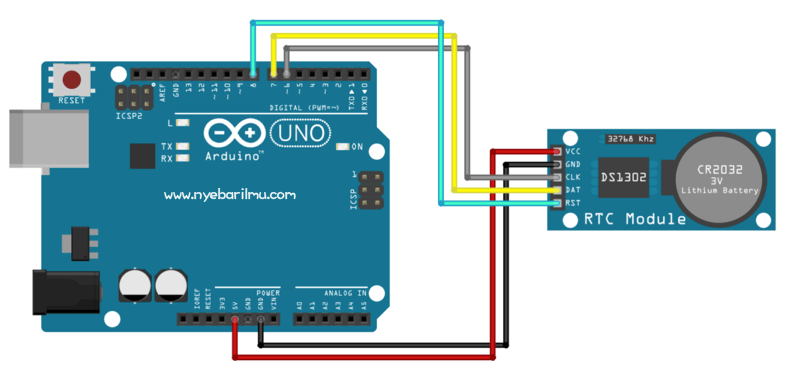
RTC DS1302 modul je komponenta koja prati realno vrijeme i datum. Sastoji se od brojila i nekoliko registara za spremanje podataka o vremenu.



Slika 3.1.3.1 Prikaz korištenog RTC modula DS1302

Za davanje točnog signala takta koristi se kristal kvarca frekvencije 32768Hz. Baterija služi za rad brojila i čuvanje sadržaja registara kada uređaj nije priključen na napajanje. Jednom kada se na RTC modulu podesi vrijeme, ono se čuva sve dok baterija traje (5-10 god.). RTC DS1302 ima 31KB NVRAM-a za pohranu podataka [7]. Komunikacija s mikroupravljačem ostvaruje se preko jednostavne sinkrone serijske komunikacije za koju je potrebno samo 3 vodiča:

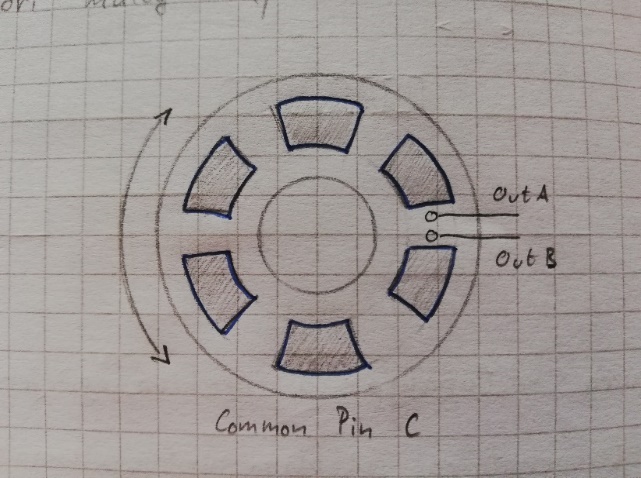
* CE (Chip Enable)
* I/O (Input/Output Data Line)
* SCLK (Serial Clock)



Slika 3.1.3.2 Prikaz spajanja RTC modula DS1302

### **3.1.4 Okretni davač**

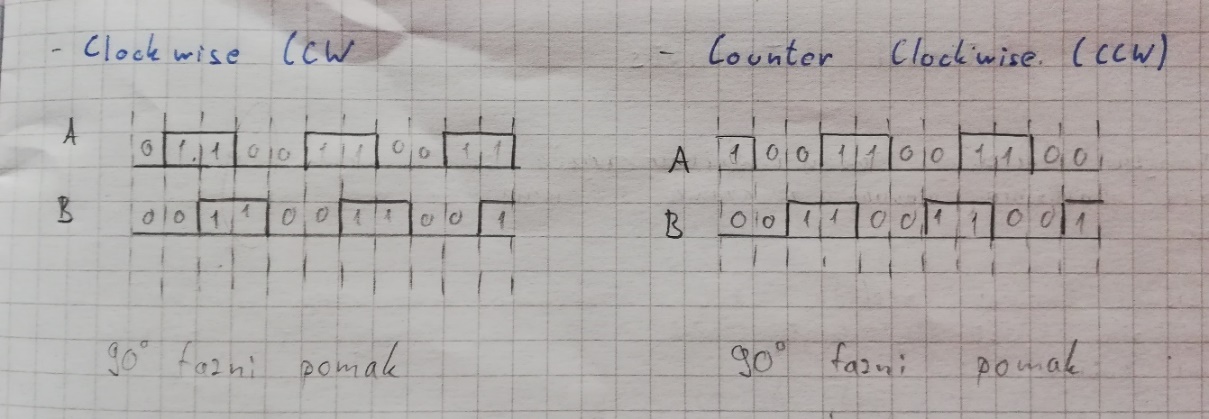
Ova komponenta je senzor koji detektira kutnu poziciju i brzinu pretvarajući rotacijski mehanički pomak u električne impulse. Moguće je detektirati i smjer okretanja uspoređujući dva impulsa koji su fazno pomaknuti za 90°. Imaju neograničen broj okretaja u oba smjera i najčešće dolaze s gumbom koji se nalazi na dnu osovine. Sama komponenta sastoji se od 3 kontakta – kontakt C (common) koji je uvijek pod naponom te 2 kontakta A i B koja kada se nađu u dodiru sa C kontaktom zatvaraju krug. Kontakti A i B su fazno pomaknuti za 90°. Ovisno o smjeru okretanja osovine, jedan od kontakata će prije doći na C kontakt [8]. Na komponenti se nalazi 5 pinova:



Slika 3.1.4.1 Pojednostavljena shema okretnog davača

* CLK - B kontakt
* DT - A kontakt
* SW - gumb
* Vcc (+5V) - napajanje
* GND – uzemljenje

U ovom projektu ova će se komponenta koristiti za upravljanje izbornikom. Okretanje osovine pomicati će kursor po zaslonu te pritiskom na tipkalo će se opcija odabrati. Na slici se vidi princip rada te kako se tumači smjer rotacije osovine



Slika 3.1.4.2 Princip rada okretnog davača

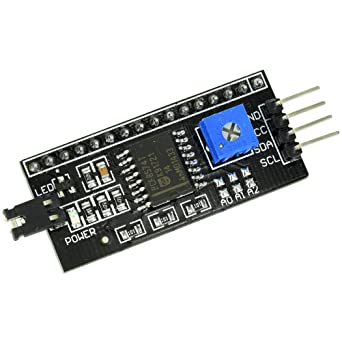
### **3.1.5 LCD zaslon**

Komponentu LCD zaslona u ovom projektu koristi se za prikaz izbornika. Veličina zaslona je 20 znakova i 4 reda (20x4) što omogućuje ispisivanje maksimalno 80 znakova u jednom trenutku. Tehnologija zaslona bazirana je na kristalima koji blokiraju svjetlost kada se kroz njih pusti električna energija [9]. Za potpunu uporabu LCD zaslona, potrebno je 16 pinova [10].



Slika 3.1.5.1 Prikaz LCD zaslona

Zbog ostalih komponenti koje će se koristiti u projektu, na LCD se može dodati adapter koji omogućava komunikaciju pomoću I2C komunikacijskog protokola. Za ovu komunikaciju su potrebna samo 2 pina te napajanje (5V) i uzemljenje. Ta dva pina su SDA (Serial Data) i SCL (Serial Clock).



Slika 3.1.5.2 I2C adapter za LCD zaslon

Kao što je prije navedeno, neki pinovi na Arduino Uno imaju više funkcija. Ove funkcije nalaze se na pinovima PC4 (A4 - SDA) i PC5 (A5 - SCL). I2C radi na master-slave principu te omogućava više uređaja na sabirnicama SDA i SCL dokle god nemaju istu adresu. Biblioteka koja je korištena za pokretanje LCD zaslona zove se LiquidCrystal\_I2C.h. Uz ovu biblioteku potrebna je još i Wire.h koji omogućuje I2C komunikaciju na Arduino pinovima A4 i A5.

### **3.1.6 Piezo zujalica**

Ova komponenta je izlazni uređaj kojem je uloga davanje jednostavnih zvučnih signala. Služi za informiranje korisnika o određenom događaju. Radi na principu obrnutog piezoelektričnog efekta – promjenjivi napon na piezoelektričnom kristalu stvara mehaničku promjenu dimenzija kristala koji će stvoriti zvuk u skladu s frekvencijom promjene napona [11].



Slika 3.1.6.1 Piezo zujalica

Piezo zujalica koja se koristi u ovom projektu ima 2 kontakta. Prvi kontakt koji je označen sa „+“ na vrhu komponente spaja se na bilo koji digitalni pin Arduina. Drugi kontakt je za uzemljenje komponente. Za upravljanje piezo zujalicom nije potrebna posebna biblioteka već se koristi naredba „tone( pin, frekvencija, trajanje tona )“. U ovom projektu, komponenta će davati signal kada se odabere neki od podizbornika te kao izvor zvuka za alarm.

## **3.2 Izrada rada**

Za početak je bilo potrebno nabaviti sve potrebne dijelove za ovaj projekt. Većina njih je nabavljena putem „ebay.com“ po relativno povoljnim cijenama. Uz potrebne komponente su nabavljene rezervne komponente u slučaju kvarova. Popis komponenti je idući:

- Arduino Uno

- LCD zaslon 20x4

- Okretni davač

- RTC modul DS1302

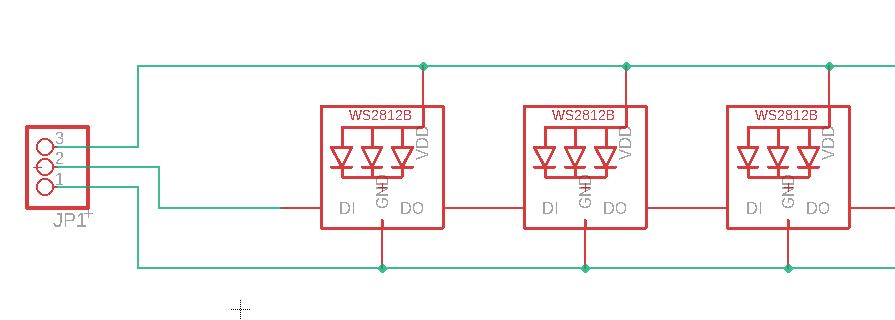
- RGB LED traka WS2812B

- Piezo zujalica

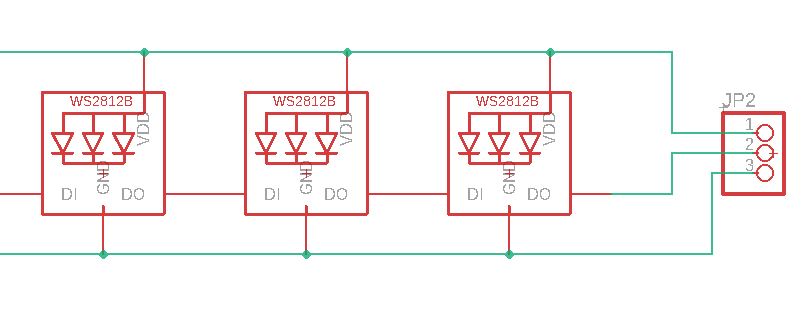
- Vodiči (M-M, M-F, F-F)

Dok su komponente bile na putu, bilo je potrebno istražiti kako pokrenuti navedene komponente. Pri tom koraku je korišten Internet. Za upravljanje RGB LED diodama WS2812B pronađena je <FastLED.h> biblioteka zbog svoje jednostavnosti. Za LCD 20x4 zaslon korištena je biblioteka <LiquidCrystal\_I2C.h> biblioteka koja je korištena na nastavi. Omogućava komunikaciju i upravljanje preko I2C serijske komunikacije. Za upravljanje RTC modulom DS1302 je korišten <DS1302RTC.h>. Nađen je dobar tutorial na GitHub koji je jako dobro objašnjavao korištenje biblioteke, upravljanje modulom te prikazivanju njegovih mogućnosti. Kada su komponente stigle, bilo je potrebno testirati njihovu funkcionalnost te početi pisati program. Započelo se sa osnovom ovog rada, satom i programom koji ga pokreće.

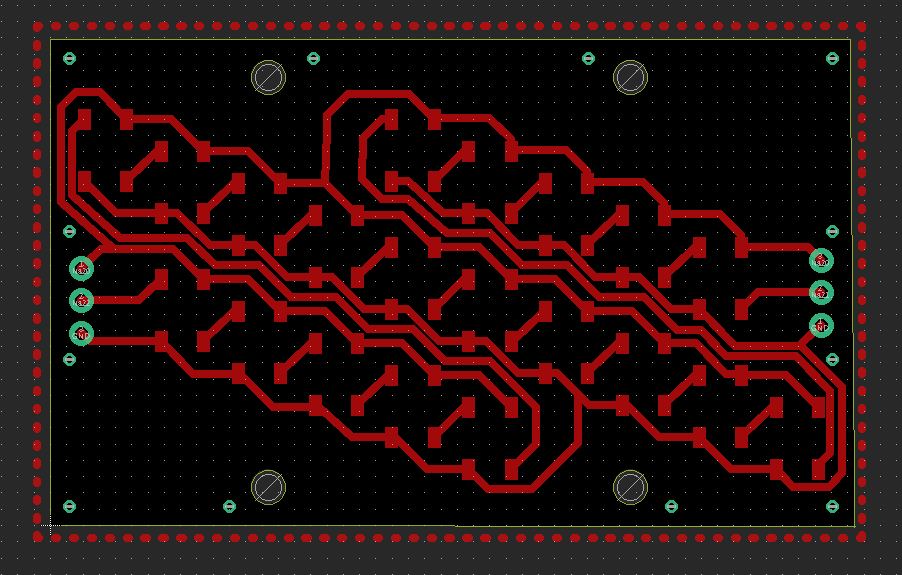
Slika 3.2.1 Djelomična električna shema spoja RGB LED dioda WS2812B na tiskanim pločicama [1]



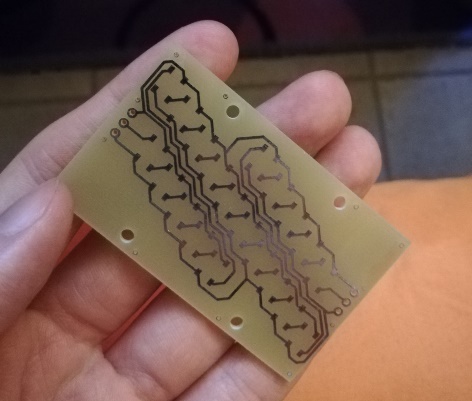
...



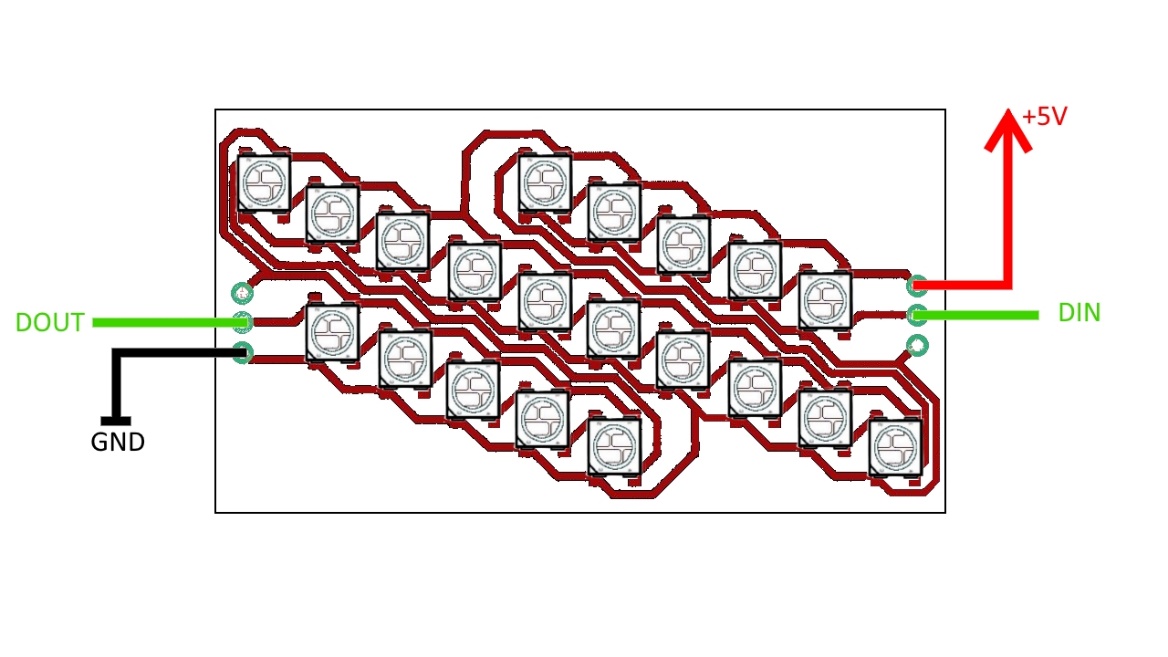
Zbog loše zalemljenog kontakta na RGB LED traci WS2812B, bilo je teško testirati funkcionalnost sata. No zbog relativno jednostavnog dijela programa koji je napisan, nije bilo problema jer je sve brzo obavljeno. Nakon vrlo uspješnog testiranja komponenti, bilo je vrijeme za izradu tiskanih pločica na koje će biti zalemljene RGB LED diode WS2812B. Te tiskane pločice su ključni dio završnog rada jer su one temelj Lixie segmenata.



Slika 3.2.1 Prikaz sheme tiskane pločice [5 (modificirano)]



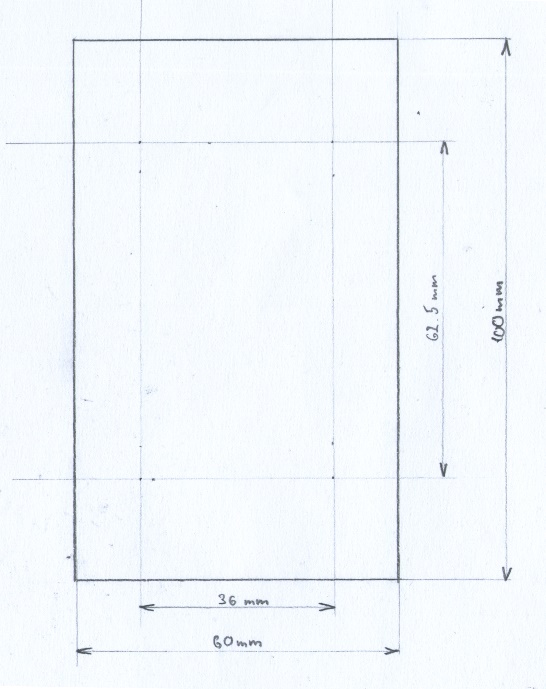
Slika 3.2.2 Prikaz jedne tiskane pločice



Slika 3.2.3 Prikaz spoja RGB LED dioda WS2812B na tiskanim pločicama

Uz tiskane pločice, bilo je potrebno izraditi akrilne pločice na kojim će biti ugravirani brojevi od 0 do 9.

Slika 3.2.4 Nacrt sa dimenzijama akrilnih pločica



Iako nije bilo sve išlo prema planu, pločice su uspješno izrađene i diode su relativno dobro zalemljene te su Lixie segmenti bili funkcionalni. Na kraju je bilo potrebno napraviti kutiju u kojoj će rad stajati. Odabrano je drvo zbog relativno jednostavne obrade.



Slika 3.2.6 Prikaz projekta sa desne strane



Slika 3.2.5 Prikaz projekta s lijeve strane

## **3.3 Konačni izgled rada**

Nakon što se program pokrene i učita do kraja, na LCD zaslonu će se prikazati poruka da je spreman za daljnje korištenje i primanje korisničkih postavki te kako pristupiti izborniku. Poruka je prikazana na slici 3.3.1. Dok je ova poruka aktivna, sat radi normalno. Pri pritisku na tipkalo ugrađeno u okretni davač, onda se mijenja.



Slika 3.3.1 Poruka koja se prikaže nakon potpune inicijalizacije programa

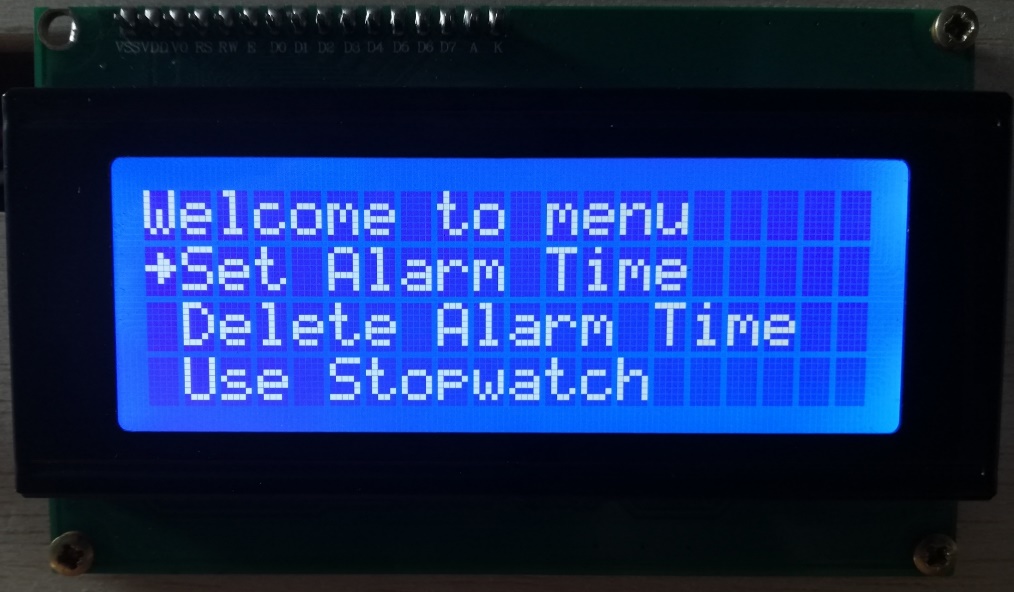
LCD zaslon briše poruku sa slike 3.3.1 te učitava prvu stranicu izbornika. Na svakoj stranici se uvijek nalazi indikator da se korisnik nalazi u izborniku te tri opcije od kojih se može birati. Slika 3.3.2 prikazuje prvu stranicu koja se učitava. Pomicanje okretnog davača mijenjamo stranice izbornika.



Slika 3.3.2 Prva stranica izbornika

Na slici 3.3.3 su prikazane ostale opcije ponuđene u izborniku. Svaku od njih se odabire pritiskom na tipkalo ugrađeno u okretni davač.

Slika 3.3.3 Prikaz ostalih opcija izbornika



Uz slike izbornika, ovdje su još dvije slike kako rad izgleda u plavoj i narančastoj boji.



Slika 3.3.4 Prikaz projekta u crvenoj boji



Slika 3.3.5 Prikaz projekta u plavoj boji

# **4. Bitni dijelovi programa**

## **4.1 Korištene biblioteke**

U nastavku su prikazane sve biblioteke korištene pri izradu programa za projekt. <DS1302RTC.h> služi za upravljanje RTC modulom, <EEPROM.h> se koristi za spremanje i učitavanje korisničkih podataka pri ponovnom pokretanju rada, <Time.h> i <TimeLib.h> su u uporabi za dodavanje dodatnih funkcija pri korištenju RTC modula. <LiquidCrystal\_I2C.h> se koristi za I2C komunikaciju sa LCD zaslonom, a potporu za I2C omogućava <Wire.h>. Biblioteka <FastLED.h> koristi se za upravljanje Lixie segmentima.

#include <DS1302RTC.h> //Za RTC modul

#include <EEPROM.h> //Za korištenje EEPROM-a

#include <Time.h> //Dodatne funkcionalnosti vremena

#include <TimeLib.h> //

#include <LiquidCrystal\_I2C.h> //I2C verzija za LCD

#include <Wire.h> //Omogućava I2C

#include <FastLED.h> //Upravljanje WS2812B diodama

Prikaz korištenih biblioteka

## **4.2 Učitavanje korisničkih postavki**

U ovom dijelu programa se učitavaju sve korisničke postavke. Varijabla „svj“ predstavlja svjetlinu segmenata. Varijable „r“, „g“ i „b“ predstavljaju vrijednosti za crvenu, zelenu i plavu boju koje te određuju kakve će boje biti segmenti. Polje varijabli x je zaduženo za pamćenje vremena postavljenog alarma.

svj = EEPROM.read(0); //Čitaju se spremljene

r = EEPROM.read(1); //vrijednosti iz EEPROM-a

g = EEPROM.read(2); //mikroupravljača ATmega328p

b = EEPROM.read(3); //

x[0] = EEPROM.read(4); //Učitavaju se vrijednosti

x[1] = EEPROM.read(5); //za svjetlinu, boju i

x[2] = EEPROM.read(6); //vrijeme za kad je alarm

x[3] = EEPROM.read(7); //postavljen

as = EEPROM.read(8); //Učitavanje ima li alarma ili ne

Učitavanje korisničkih postavki

## **4.3 Inicijalne postavke programa**

Dok se sve na Arduinu učitava i pokreće, na LCD zaslonu će se pojaviti poruke prikazane u nastavku. Sa ovime završava inicijalno postavljanje i sat počinje da radom.

lcd.init();

lcd.*begin*(20,4);

lcd.backlight();

lcd.setCursor(0,0);

lcd.print("Hello, Mate!");

lcd.createChar(1, arrow);

delay(2000);

lcd.setCursor(0,0);

lcd.print("Welcome to menu");

delay(2000);

Serial.print("Setup complete!");

Serial.print("\n");

Inicijalne postavke s kojima završava postavljanje programa i kreće sat sa radom

## **4.4 Čitanje vremena s RTC modula**

U nastavku je prikazan proces dobivanja vremena. Dobiveni brojevi se spremaju u zasebne varijable sa imenima koje predstavljaju sate (h), minute (m) i sekunde (s). Ove varijable su zapravo polja. Zato što se izvorni podaci dobivaju kao dvoznamenkasti brojevi, postoji kod za rastavu na zasebne segmente. To se radi pomoću koda:

for (i = 0; i < 2; i++){

if (i == 0){

h [i] = n / 10;

}

if (i == 1){

h [i] = n % 10;

}

}

…

Za „h [i] = n / 10“ dobiva se prva znamenka sati. Zbog oblika varijable (int tip varijable), ne moramo se brinuti za decimalni dio broja jer se on ne sprema. Za „h [i] = n % 10“ dobiva se druga znamenka sati. Operacija % (MOD) daje ostatak cjelobrojnog dijeljenja. Taj rezultat je druga znamenka sati. Isti princip se primjenjuje za minute i sekunde.

tmElements\_t tm;

RTC.read(tm);

n = tm.Hour;

for (i=0; i<2; i++){

if(i == 0){

h[i] = n / 10;

}

if(i == 1){

h[i] = n % 10;

}

}

n = tm.Minute;

for (i=0; i<2; i++){

if(i == 0){

m[i] = n / 10;

}

if(i == 1){

m[i] = n % 10;

}

}

n = tm.Second;

for (i=0; i<2; i++){

if(i == 0){

s[i] = n / 10;

}

if(i == 1){

s[i] = n % 10;

}

}

Dobavljanje vremena sa RTC modula

## **4.5 Određivanje brojeva koji će se prikazati na Lixie segmentima**

U nastavku je prikazana jedna vrsta dekodera za brojeve koji su dobiveni u prijašnjem dijelu koda. Služi za određivanje što će se ispisati na kojem segmentu. Princip je vrlo jednostavan.

1. Provjerava se koji je broj dobiven od prije. Za određeni broj, program će otići u jedan od ponuđenih if funkcija.
2. Kada program uđe u if funkciju, prvo će ugasiti sve RGB LED komponente jednog segmenta. Nakon toga u varijablu jn (gdje n predstavlja broj segmenta) stavlja broj. Taj broj određuje prvu RGB LED komponentu koja će se upaliti će se još kasnije koristiti.

Za sad, još nema prikaza brojeva. Ovaj kod je samo logika koja određuje koji će se broj prikazati. Isti princip dekodiranja je korišten za ostale segmente minuta i sekundi.

if(h[0] != h1p){ h1p = h[0];

if(h[0] == 0){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 103; //0

}

if(h[0] == 1){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 104; //1

}

if(h[0] == 2){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 102; //2

}

if(h[0] == 3){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 100; //3

}

if(h[0] == 4){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 108; //4

}

if(h[0] == 5){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 106; //5

}

if(h[0] == 6){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 105; //6

}

if(h[0] == 7){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 107; //7

}

if(h[0] == 8){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 109; //8

}

if(h[0] == 9){

for(z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0); FastLED.show();

}

j1 = 101; //9

}

}

Prikazuje odabir broja koji će se ispisati na prvom segmentu

## **4.6 Ispisivanje vremena na Lixie segmentima**

Pomoću dijela programa prikazanog u nastavku uključuju se Lixie segmenti. Za svaku brojku su namijenjene 2 RGB LED komponente.

Ova naredba određuje boju RGB LED komponenti

„leds [j1] = CRGB (r, g, b);“

Ova naredba daje naredbu da se određena RGB LED komponenta uključi.

„FastLED.show();“

Ova naredba uključuje u RGB LED komponentu koja se nalazi u istom retku sa prvom komponentom.

„leds [j1+10] = CRGB (r, g, b);“

void showing(){

//H[0]

leds[j1] = CRGB(r, g, b);

FastLED.show();

leds[j1+10] = CRGB(r, g, b);

FastLED.show();

//H[1]

leds[j2] = CRGB(r, g, b);

FastLED.show();

leds[j2+10] = CRGB(r, g, b);

FastLED.show();

//M[0]

leds[j3] = CRGB(r, g, b);

FastLED.show();

leds[j3+10] = CRGB(r, g, b);

FastLED.show();

//M[1]

leds[j4] = CRGB(r, g, b);

FastLED.show();

leds[j4+10] = CRGB(r, g, b);

FastLED.show();

//S[0]

leds[j5] = CRGB(r, g, b);

FastLED.show();

leds[j5+10] = CRGB(r, g, b);

FastLED.show();

//S[1]

leds[j6] = CRGB(r, g, b);

FastLED.show();

leds[j6+10] = CRGB(r, g, b);

FastLED.show();

}

Dio programa koji ispisuje brojeve na Lixie segmentima

## **4.7 Ulazak u izbornik**

U nastavku je prikazan dio programa koji, kada se zadovolji, poziva funkciju „menu()“. Ta funkcija predstavlja izbornik programa. Ona će se pokrenuti ako korisnik pritisne tipkalo ugrađeno u okretni davač. Ovdje se samo provjerava je li tipkalo pritisnuto.

if(digitalRead(10) == 0){

menu();

}

Prikazuje ulazak u izbornik

## **4.8 Alarmi**

Ovaj dio programa se odnosi na alarme. U nastavku je prikazan samo jedan od tri alarma. Prva if funkcija služi kao provjera je li došlo vrijeme za alarm. Varijabla „as“ provjerava postoji li uopće namješteni alarm. Varijabla „a“ određuje kakav će biti zvuk alarma. For petlja prvo gasi sve segmente te je zatim druga for petlja odgovorna za kreiranje zvuka i vizualnih efekata. Kada alarm započne, Lixie segmenti će se uključivati i isključivati u crvenoj boji. Broj koji će biti prikazan je 0.

if(as != 0 && x[0] == h[0] && x[1] == h[1] && x[2] == m[0] && x[3] == m[1]){

if(a == 1){

l=0;

do{

lcd.print("Get up!");

tone(po, 500, 500);

delay(500);

noTone(po);

lcd.clear();

delay(500);

l++;

if(l==20){

as=0;

EEPROM.write(8, as);

}

}while(as==1);

}

if(a == 2){

l=0;

do{

lcd.print("Get up!");

tone(po, 500, 500);

delay(100);

noTone(po);

lcd.clear();

delay(100);

l++;

if(l==20){

as=0;

EEPROM.write(8, as);

}

}while(as==1);

}

}

Prikazuje dio programa koji određuje alarme

## **4.9 Izbornik**

U nastavku je prikazan dio koda koji pokreće izbornik. Upravljanje izbornikom se obavlja putem okretnog davača. Pomak za jedan u desno spušta strelicu koja pokazuje trenutno odabranu opciju. Pomak za jedan u lijevo podiže strelicu. U neki ruku bi se moglo reći da svaki pomak okretnog davača mijenja stranicu. Varijabla „counter“ je odgovorna za utvrđivanje stranice. Opcije izbornika se određuju pomoću tipkala ugrađenog u okretni davač. Kako bi korisnik znao da je odabrao neku opciju, program će dati auditivni odgovor pomoću piezo zujalice te porukom na LCD zaslonu koja glasi: „Just a second!“. Na samoj slici su prikazane samo dvije stranice izbornika.

void menu(){

if((last\_counter > counter) || (last\_counter < counter) || pushed){

Ready = 1;

if(submenu == 0){

do{

if(counter != lcounter){

clr = 0;

lcounter = counter;

}

if(0 < counter && counter < 4){

if(clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0,0);

lcd.print("Welcome to menu");

lcd.setCursor(0,1);

lcd.write(1);

lcd.print("Set Brightness");

lcd.setCursor(1,2);

lcd.print("Set Color");

lcd.setCursor(1,3);

lcd.print("Set Alarm Tone");

page = 1;

delay(200);

if(digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Just a second!");

delay(1000);

submenu=1;

counter=2;

}

}

if(4 < counter && counter < 8){

if(clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0,0);

lcd.print("Welcome to menu");

lcd.setCursor(0,1);

lcd.write(1);

lcd.print("Set Color");

lcd.setCursor(1,2);

lcd.print("Set Alarm Tone");

lcd.setCursor(1,3);

lcd.print("Set Alarm Time");

page=2;

delay(200);

if(digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Just a second!");

delay(1000);

submenu=2;

counter=2;

}

}

...

Prikazuje dio programa odgovornog za izbornik

## **4.10 Prekid za okretni davač**

U nastavku je prikazan prekidni dio programa koji poveća ili smanjuje vrijednost varijable „counter“ ovisno o smjeru u koji se okreće. Zna se u kojem smjeru se okreće na principu koji je opisan kod komponente. Ukratko, ovisno u kojem smjeru se okreće, jedan od pinova A i B će prije doći na zajednički kontakt C.

ISR(PCINT0\_vect){

clk\_State = (PINB & B00000001);

dt\_State = (PINB & B00000010);

if (clk\_State != Last\_State){

if (dt\_State != clk\_State){

counter ++;

}else{

counter --;

}

}

Last\_State = clk\_State;

}

Prikazuje prekidni program za okretni davač

# **5. Zaključak**

Sam koncept projektnog zadatka se činio vrlo jednostavan. Mislio sam da ću bez problema riješiti projekt u dva do tri mjeseca maksimalno. I evo me ovdje, sa završenim projektom dva tjedna prije predaje. Uz nekoliko poteškoća, projekt je uspješno završen i moram priznati da sam poprilično zadovoljan. Najviše problema sam imao sa lemljenjem. U školi po programu nisam učio pa sam to morao proučiti na svoju ruku. Prve izrađene tiskane pločice su bile… ne baš kvalitetne. Moguće da su imale stop-lak koji je sprječavao oksidiranje i uz to, lemljenje. Zato je prva pločica izgledala katastrofalno. No nakon što su izrađene druge modificirane pločice, sve je više-manje sjelo na svoje mjesto. Ograničenje je jedino bilo moje slobodno vrijeme zbog ostalih školskih obaveza. Ova školska godina je općenito bila kaotična. Unatoč tome, naučio sam mnogo. Ne samo kako izbjeći potpuni slom živaca nego i stvari koje će mi općenito pomoći u životu. I pod time mislim općenito na moje četiri godine u Tehničkoj školi Ruđera Boškovića. Možda je vrlo teška i zahtjevna, ali stvarno sam naučio mnogo od čega je većina zapravo ono što me zanima. Jako mi je drago što nisam otišao u neku od gimnazija. Ovako sam učio stvari koje me zanimaju. Neke od znanja koje sam stekao su korištenje programa za kreiranje i modificiranje tiskanih pločica, polu-profesionalno komuniciranje sa raznim firmama, ispravno pisanje dokumentacije za projekte, korištenje komponenti koje nisu bile pokrivene na nastavi i mnogo drugih stvari. U svakom slučaju će biti korisno. Jedino mi je žao što nisam imao priliku raditi sa 3D printerom. To bi bio odličan način za izraditi filtere svjetlosti i dodatno poboljšati ovaj projektni zadatak.

# **6. Literatura**

1. Connor Nishijima. GitHub, Lixie-hardware*,* 23. siječnja, 2016., <https://github.com/connornishijima/Lixie-hardware> , (Pregledano u lipnju 2019.)
2. Arduino, Arduino Uno Rev3, <https://store.arduino.cc/arduino-uno-rev3> , (Pregledano 24.3.2020.)
3. Elecrom, [Embedded Electronics Blog](https://www.elecrom.com/), Introduction to Arduino UNO (uses AVR ATmega328), <https://www.elecrom.com/introduction-arduino-uno-uses-avr-atmega328/> , 27. siječnja, 2017. (Pregledano 24.3.2020.)
4. Dumančić Z., Prezentacija s predavanja 0.2 Ugradbeni računalni sustavi – I2C komunikacija, 11. studenog, 2019. (Pregledano u studenom, 2019.)
5. Lisa Horn, Ponoko, Lighting Up Sales With The Lixie II, 12. prosinca, 2019. <https://www.ponoko.com/blog/maker-success-stories/led-nixie-clock-lixie-2/> (Pregledano 15.5.2020.)
6. WS2812B*,* <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf> , (Pregledano 4.5.2020.)
7. DS1302 Trickle-Charge Timekeeping Chip., <https://datasheets.maximintegrated.com/en/ds/DS1302.pdf> ,(Pregledano 24.3.2020.)
8. Dumančić Z., Prezentacija s predavanja 0.5 Ugradbeni računalni sustavi – Okretni davač, 31. siječnja, 2020. (Pregledano u veljači, 2020.)
9. Tarun Agarwal, Construction and Working Principle of LCD Display*,*  <https://www.elprocus.com/ever-wondered-lcd-works/> , (Pregledano 27.3.2020.)
10. LCD-020N004L*,*  <https://www.vishay.com/docs/37314/lcd020n004l.pdf> , 1. siječnja, 2019., (Pregledano 24.3.2020.)
11. Piezoelectronic Buzzers*.* <https://www.mouser.com/datasheet/2/400/ef532_ps-13444.pdf> ,svibanj, 2011., (Pregledano 24.3.2020.)

# **7. Prilozi**

## **7.1 Cjelokupni programski kod**

#include <DS1302RTC.h> //Uključivanje potrebnih biblioteka

#include <EEPROM.h>

#include <Time.h>

#include <TimeLib.h>

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

#include <FastLED.h>

#define po 7

#define push 10

#define DS1302\_GND\_PIN 5

#define DS1302\_VCC\_PIN 6

#define LED\_PIN A0

#define NUM\_LEDS 120

LiquidCrystal\_I2C lcd(0x3F, 20, 4); //Inicijalizacija LCD zaslona

//Kreiranje znaka strelice

uint8\_t arrow[8] = { 0x0, 0x04, 0x06, 0x1f, 0x06, 0x04, 0x00, 0x00 }

int w = 0;

int v = 0;

int i = 0;

int j1 = 0; //Varijable koje služe kao

int j2 = 0; //privremeni spremnici

int j3 = 0; //pomoću kojih se dekodira

int j4 = 0; //koji broj se treba

int j5 = 0; //prikazati na segmentu

int j6 = 0; //

int z1 = 0;

int z2 = 20;

int z3 = 40;

int z4 = 60;

int z5 = 80;

int z6 = 100;

int n = 0;

int h[2]; //Varijable koje upravljaju

int m[2]; //segmentima

int s[2]; //

int h1p = 0; //

int h2p = 0; //

int m1p = 0; //

int m2p = 0; //

int s1p = 0; //

int s2p = 0; //

int svj = 255; //Varijabla koja upravlja svjetlinom WS2812B dioda

int r = 0; //Varijabla za crvenu boju

int g = 0; //Varijabla za zelenu boju

int b = 0; //Varijabla za plavu boju

int lcounter = 0;

int counter = 0;

int page = 1;

int Ready = 1;

int submenu = 0;

int last\_counter = 0;

bool clk\_State;

bool Last\_State;

bool dt\_State;

int pushed = 0;

int ch = 0; //Varijabla za alarm (sati)

int cm = 0; //Varijabla za alarm (minute)

int x[4];

int a = 0; //Varijabla koja određuje

int as = 0; //ima li alarma ili ne

int l = 0;

int cler = 0;

int clr = 0;

int zet = 0;

int sh = 0;

int sm = 0;

int ss = 0;

int sh1 = 0;

int sh2 = 0;

int sm1 = 0;

int sm2 = 0;

int ss1 = 0;

int ss2 = 0;

DS1302RTC RTC(2, 3, 4); //Inicijalizacija RTC modula na pinovima 2, 3, 4

CRGB leds[NUM\_LEDS]; //Inicijalizacija WS2812B dioda

void setup() {

svj = EEPROM.read(0); //Čitaju se spremljene

r = EEPROM.read(1); //vrijednosti iz EEPROM-a

g = EEPROM.read(2); //mikroupravljača ATmega328p

b = EEPROM.read(3); //

x[0] = EEPROM.read(4); //Učitavaju se vrijednosti

x[1] = EEPROM.read(5); //za svjetlinu, boju i

x[2] = EEPROM.read(6); //vrijeme za kad je alarm

x[3] = EEPROM.read(7); //postavljen

as = EEPROM.read(8);

PCICR |= (1 << PCIE0);

PCMSK0 |= (1 << PCINT0);

PCMSK0 |= (1 << PCINT1);

DDRB &= B11111100;

Last\_State = (PINB & B00000001);

pinMode(push, INPUT\_PULLUP);

FastLED.addLeds<WS2812, LED\_PIN, GRB>(leds, NUM\_LEDS);

Serial.begin(9600);

digitalWrite(DS1302\_GND\_PIN, LOW);

pinMode(DS1302\_GND\_PIN, OUTPUT);

digitalWrite(DS1302\_VCC\_PIN, HIGH);

pinMode(DS1302\_VCC\_PIN, OUTPUT);

Serial.print("RTC activated!\n");

if (RTC.haltRTC())

Serial.print("Clock stopped!\n");

else

Serial.print("Clock working.\n");

if (RTC.writeEN())

Serial.print("Write allowed\n.");

else

Serial.print("Write protected.\n");

setSyncProvider(RTC.get);

if (timeStatus() == timeSet)

Serial.print(" Ok!\n");

else

Serial.print(" FAIL!\n");

lcd.init();

lcd.begin(20, 4);

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Hello, Mate!");

lcd.createChar(1, arrow);

delay(2000);

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

delay(2000);

Serial.print("Setup complete!");

Serial.print("\n");

}

void loop(){

counter = 2;

lcd.setCursor(0, 0);

lcd.print("Standby & Ready!");

lcd.setCursor(0, 1);

lcd.print("Press button to open");

lcd.setCursor(0, 2);

lcd.print("the menu. Clock will");

lcd.setCursor(0, 3);

lcd.print("stop when menu opens");

FastLED.setBrightness(svj);

tmElements\_t tm;

RTC.read(tm);

n = tm.Hour;

for (i = 0; i<2; i++){

if (i == 0){

h[i] = n / 10;

}

if (i == 1){

h[i] = n % 10;

}

}

n = tm.Minute;

for (i = 0; i<2; i++){

if (i == 0){

m[i] = n / 10;

}

if (i == 1){

m[i] = n % 10;

}

}

n = tm.Second;

for (i = 0; i<2; i++){

if (i == 0){

s[i] = n / 10;

}

if (i == 1){

s[i] = n % 10;

}

}

if(w==0){

h[0] = 0;

h[1] = 0;

m[0] = 0;

m[1] = 0;

s[0] = 0;

s[1] = 0;

j1 = 103;

j2 = 83;

j3 = 63;

j4 = 43;

j5 = 23;

j6 = 3;

w = 1;

}

if (h[0] != h1p){

h1p = h[0];

//Hours

if (h[0] == 0){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 103;

//0

}

if (h[0] == 1){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 104;

//1

}

if (h[0] == 2){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 102;

//2

}

if (h[0] == 3){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 100;

//3

}

if (h[0] == 4){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 108;

//4

}

if (h[0] == 5){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 106;

//5

}

if (h[0] == 6){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 105;

//6

}

if (h[0] == 7){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 107;

//7

}

if (h[0] == 8){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 109;

//8

}

if (h[0] == 9){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 101;

//9

}

}

if (h[1] != h2p){

h2p = h[1];

//Hours

if (h[1] == 0){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 83;

//0

}

if (h[1] == 1){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 84;

//1

}

if (h[1] == 2){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 82;

//2

}

if (h[1] == 3){

for (z2 = 20; z2 < 40; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 80;

//3

}

if (h[1] == 4){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 88;

//4

}

if (h[1] == 5){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 86;

//5

}

if (h[1] == 6){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 85;

//6

}

if (h[1] == 7){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 87;

//7

}

if (h[1] == 8){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 89;

//8

}

if (h[1] == 9){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 81;

//9

}

}

if (m[0] != m1p){

m1p = m[0];

//Minutes

if (m[0] == 0){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 63;

//0

}

if (m[0] == 1){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 64;

//1

}

if (m[0] == 2){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 62;

//2

}

if (m[0] == 3){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 60;

//3

}

if (m[0] == 4){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 68;

//4

}

if (m[0] == 5){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 66;

//5

}

if (m[0] == 6){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 65;

//6

}

if (m[0] == 7){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 67;

//7

}

if (m[0] == 8){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 69;

//8

}

if (m[0] == 9){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 61;

//9

}

}

if (m[1] != m2p){

m2p = m[1];

//Minutes

if (m[1] == 0){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 43;

//0

}

if (m[1] == 1){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 44;

//1

}

if (m[1] == 2){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 42;

//2

}

if (m[1] == 3){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 40;

//3

}

if (m[1] == 4){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 48;

//4

}

if (m[1] == 5){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 46;

//5

}

if (m[1] == 6){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 45;

//6

}

if (m[1] == 7){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 47;

//7

}

if (m[1] == 8){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 49;

//8

}

if (m[1] == 9){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 41;

//9

}

}

if (s[0] != s1p){

s1p = s[0];

//Seconds

if (s[0] == 0){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 23;

//0

}

if (s[0] == 1){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 24;

//1

}

if (s[0] == 2){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 22;

//2

}

if (s[0] == 3){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 20;

//3

}

if (s[0] == 4){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 28;

//4

}

if (s[0] == 5){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 26;

//5

}

if (s[0] == 6){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 25;

//6

}

if (s[0] == 7){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 27;

//7

}

if (s[0] == 8){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 29;

//8

}

if (s[0] == 9){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 21;

//9

}

}

if (s[1] != s2p){

s2p = s[1];

//Seconds

if (s[1] == 0){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 3;

//0

}

if (s[1] == 1){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 4;

//1

}

if (s[1] == 2){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 2;

//2

}

if (s[1] == 3){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 0;

//3

}

if (s[1] == 4){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 8;

//4

}

if (s[1] == 5){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 6;

//5

}

if (s[1] == 6){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 5;

//6

}

if (s[1] == 7){

for (z6 = 0; z6 < 20; z6++){

leds[z6

] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 7;

//7

}

if (s[1] == 8){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 9;

//8

}

if (s[1] == 9){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 1;

//9

}

}

showing();

if (digitalRead(10) == 0){

menu();

}

x[0] = EEPROM.read(4);

x[1] = EEPROM.read(5);

x[2] = EEPROM.read(6);

x[3] = EEPROM.read(7);

if(as != 0 && x[0] == h[0] && x[1] == h[1] && x[2] == m[0] && x[3] == m[1]){

Serial.print("Alarm detected!");

if(a == 1){

l=0;

do{

lcd.print("Get up!");

tone(po, 500, 500);

delay(500);

noTone(po);

lcd.clear();

delay(500);

l++;

if(l==20){

as=0;

EEPROM.write(8, as);

}

}while(as==1);

}

if(a == 2){

l=0;

do{

lcd.print("Get up!");

tone(po, 500, 500);

delay(100);

noTone(po);

lcd.clear();

delay(100);

l++;

if(l==20){

as=0;

EEPROM.write(8, as);

}

}while(as==1);

}

}

}

void menu(){

if ((last\_counter > counter) || (last\_counter < counter) || pushed){

Ready = 1;

if (submenu == 0){

do{

if (counter != lcounter){

clr = 0;

lcounter = counter;

}

if (0 < counter && counter < 4){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Set Brightness");

lcd.setCursor(1, 2);

lcd.print("Set Color");

lcd.setCursor(1, 3);

lcd.print("Set Alarm Tone");

page = 1;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 1;

counter = 2;

}

}

if (4 < counter && counter < 8){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Set Color");

lcd.setCursor(1, 2);

lcd.print("Set Alarm Tone");

lcd.setCursor(1, 3);

lcd.print("Set Alarm Time");

page = 2;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 2;

counter = 2;

}

}

if (8 < counter && counter < 12){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Set Alarm Tone");

lcd.setCursor(1, 2);

lcd.print("Set Alarm Time");

lcd.setCursor(1, 3);

lcd.print("Delete Alarm Time");

page = 3;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 3;

counter = 2;

}

}

if (12 < counter && counter < 16){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Set Alarm Time");

lcd.setCursor(1, 2);

lcd.print("Delete Alarm Time");

lcd.setCursor(1, 3);

lcd.print("Use Stopwatch");

page = 4;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 4;

counter = 2;

}

}

if (16 < counter && counter < 20){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Delete Alarm Time");

lcd.setCursor(1, 2);

lcd.print("Use Stopwatch");

lcd.setCursor(1, 3);

lcd.print("Set & Use Timer");

page = 5;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 5;

counter = 2;

}

}

if (20 < counter && counter < 24){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Use Stopwatch");

lcd.setCursor(1, 2);

lcd.print("Set & Use Timer");

lcd.setCursor(1, 3);

lcd.print("EXIT");

page = 6;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 6;

counter = 2;

}

}

if (24 < counter && counter < 28){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(1, 1);

lcd.print("Use Stopwatch");

lcd.setCursor(0, 2);

lcd.write(1);

lcd.print("Set & Use Timer");

lcd.setCursor(1, 3);

lcd.print("EXIT");

page = 7;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 7;

counter = 2;

return;

}

}

if (28 < counter && counter < 32){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(1, 1);

lcd.print("Use Stopwatch");

lcd.setCursor(1, 2);

lcd.print("Set & Use Timer");

lcd.setCursor(0, 3);

lcd.write(1);

lcd.print("EXIT");

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

return;

}

}

} while (submenu == 0);

}

if (submenu == 1){

do{

if (counter != lcounter){

clr = 0;

lcounter = counter;

}

if (0 < counter && counter < 4){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set brightness");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("25%");

lcd.setCursor(1, 2);

lcd.print("50%");

lcd.setCursor(1, 3);

lcd.print("75%");

page = 1;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

svj = 63;

EEPROM.write(0, svj);

submenu = 0;

counter = 2;

}

}

if (4 < counter && counter < 8){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set brightness");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("50%");

lcd.setCursor(1, 2);

lcd.print("75%");

lcd.setCursor(1, 3);

lcd.print("100%");

page = 2;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

svj = 127;

EEPROM.write(0, svj);

submenu = 0;

counter = 2;

}

}

if (8 < counter && counter < 12){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set brightness");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("75%");

lcd.setCursor(1, 2);

lcd.print("100%");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 3;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

svj = 191;

EEPROM.write(0, svj);

submenu = 0;

counter = 2;

}

}

if (12 < counter && counter < 16){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set brightness");

lcd.setCursor(1, 1);

lcd.print("75%");

lcd.setCursor(0, 2);

lcd.write(1);

lcd.print("100%");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 4;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

svj = 255;

EEPROM.write(0, svj);

submenu = 0;

counter = 2;

}

}

if (16 < counter && counter < 20){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set brightness");

lcd.setCursor(1, 1);

lcd.print("75%");

lcd.setCursor(1, 2);

lcd.print("100%");

lcd.setCursor(0, 3);

lcd.write(1);

lcd.print("BACK");

page = 5;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

}

}

} while (submenu == 1);

}

if (submenu == 2){

do{

if (counter != lcounter){

clr = 0;

lcounter = counter;

}

if (0 < counter && counter < 4){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Color");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Red");

lcd.setCursor(1, 2);

lcd.print("Green");

lcd.setCursor(1, 3);

lcd.print("Blue");

page = 1;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

r = 255;

g = 0;

b = 0;

EEPROM.write(1, r);

EEPROM.write(2, g);

EEPROM.write(3, b);

submenu = 0;

counter = 2;

}

}

if (4 < counter && counter < 8){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Color");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Green");

lcd.setCursor(1, 2);

lcd.print("Blue");

lcd.setCursor(1, 3);

lcd.print("Yellow");

page = 2;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

r = 0;

g = 255;

b = 0;

EEPROM.write(1, r);

EEPROM.write(2, g);

EEPROM.write(3, b);

submenu = 0;

counter = 2;

}

}

if (8 < counter && counter < 12){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Color");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Blue");

lcd.setCursor(1, 2);

lcd.print("Orange");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 3;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

r = 0;

g = 0;

b = 255;

EEPROM.write(1, r);

EEPROM.write(2, g);

EEPROM.write(3, b);

submenu = 0;

counter = 2;

}

}

if (12 < counter && counter < 16){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Color");

lcd.setCursor(1, 1);

lcd.print("Blue");

lcd.setCursor(0, 2);

lcd.write(1);

lcd.print("Orange");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 4;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

r = 255;

g = 140;

b = 0;

EEPROM.write(1, r);

EEPROM.write(2, g);

EEPROM.write(3, b);

submenu = 0;

counter = 2;

}

}

if (16 < counter && counter < 20){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Color");

lcd.setCursor(1, 1);

lcd.print("Blue");

lcd.setCursor(1, 2);

lcd.print("Orange");

lcd.setCursor(0, 3);

lcd.write(1);

lcd.print("BACK");

page = 5;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

return;

}

}

} while (submenu == 2);

}

if (submenu == 3){

do{

if (counter != lcounter){

clr = 0;

lcounter = counter;

}

if (0 < counter && counter < 4){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Alarm Tone");

lcd.setCursor(0, 1);

lcd.write(1);

lcd.print("Alarm 1");

lcd.setCursor(1, 2);

lcd.print("Alarm 2");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 1;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

a = 1;

submenu = 0;

counter = 2;

}

}

if (4 < counter && counter < 8){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Alarm Tone");

lcd.setCursor(1, 1);

lcd.print("Alarm 1");

lcd.setCursor(0, 2);

lcd.write(1);

lcd.print("Alarm 2");

lcd.setCursor(1, 3);

lcd.print("BACK");

page = 2;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

a = 2;

submenu = 0;

counter = 2;

}

}

if (8 < counter && counter < 12){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Set Alarm Tone");

lcd.setCursor(1, 1);

lcd.print("Alarm 1");

lcd.setCursor(1, 2);

lcd.print("Alarm 2");

lcd.setCursor(0, 3);

lcd.write(1);

lcd.print("BACK");

page = 3;

delay(200);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

return;

}

}

} while (submenu == 3);

}

if (submenu == 4){

do{

delay(1000);

ch = 0;

cm = 0;

int sh = 0;

int sm = 0;

int sh1 = 0;

int sh2 = 0;

int sm1 = 0;

int sm2 = 0;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Set Alarm Time");

if (sh1 == 0 && sh2 == 0){

do{

lcd.setCursor(0, 1);

lcd.print("Hours:");

if (sh >= 0 && sh <= 9){

lcd.setCursor(0, 2);

lcd.print("0");

lcd.setCursor(1, 2);

lcd.print(sh);

}

delay(100);

if (sh >= 10 && sh <= 23){

lcd.setCursor(0, 2);

lcd.print(sh);

}

delay(100);

if (ch == 0){

if (counter > last\_counter){

sh++;

last\_counter = counter;

}

delay(100);

if (counter < last\_counter){

sh--;

last\_counter = counter;

}

delay(100);

if (sh >= 24){

lcd.setCursor(0, 2);

lcd.print("00");

sh = 0;

}

if (sh < 0){

sh = 23;

}

delay(100);

if (digitalRead(10) == 0){

sh1 = sh / 10;

sh2 = sh % 10;

EEPROM.write(4, sh1);

EEPROM.write(5, sh2);

ch = 1;

last\_counter = 0;

counter = 0;

}

}

}while (ch == 0);

}

if (sm1 == 0 && sm2 == 0){

do{

lcd.setCursor(0, 1);

lcd.print("Minutes:");

if (sm >= 0 && sm <= 9){

lcd.setCursor(0, 2);

lcd.print("0");

lcd.setCursor(1, 2);

lcd.print(sm);

}

delay(100);

if (sm >= 10 && sm <= 59){

lcd.setCursor(0, 2);

lcd.print(sm);

}

delay(100);

if (cm == 0){

if (counter > last\_counter){

sm++;

last\_counter = counter;

}

delay(100);

if (counter < last\_counter){

sm--;

last\_counter = counter;

}

delay(100);

if (sm >= 60){

lcd.setCursor(0, 2);

lcd.print("00");

sm = 0;

}

if (sm < 0){

sm = 59;

}

if (digitalRead(10) == 0){

sm1 = sm / 10;

sm2 = sm % 10;

EEPROM.write(6, sm1);

EEPROM.write(7, sm2);

cm = 1;

last\_counter = 0;

counter = 0;

}

}

} while (cm == 0);

}

as = 1;

EEPROM.write(8, as);

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 2);

lcd.print("Alarm Set!");

delay(1000);

submenu = 0;

counter = 2;

} while (submenu == 4);

}

if (submenu == 5){

do{

if (counter != lcounter){

clr = 0;

lcounter = counter;

}

if (0 < counter && counter < 5){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.print("Delete Alarm Time?");

lcd.setCursor(0, 2);

lcd.write(1);

lcd.print("YES");

lcd.setCursor(1, 3);

lcd.print("NO");

delay(300);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

lcd.clear();

lcd.print("Alarm Time Deleted");

delay(1000);

submenu = 0;

counter = 2;

EEPROM.write(4, 0);

EEPROM.write(5, 0);

EEPROM.write(6, 0);

EEPROM.write(7, 0);

}

}

if (5 < counter && counter < 10){

if (clr == 0){

lcd.clear();

clr = 1;

}

lcd.setCursor(0, 0);

lcd.print("Welcome to menu");

lcd.setCursor(0, 1);

lcd.print("Delete Alarm Time?");

lcd.setCursor(1, 2);

lcd.print("YES");

lcd.setCursor(0, 3);

lcd.write(1);

lcd.print("NO");

delay(300);

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

return;

}

}

} while (submenu == 5);

}

if (submenu == 6){

int o = 0;

int h = 0;

int e = 10;

int r[12] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

for (v = 0; v<120; v++){

leds[v] = CRGB(0, 0, 0);

FastLED.show();

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Stopwatch starts in:");

lcd.setCursor(0, 1);

lcd.print("3...");

delay(1000);

lcd.print("2...");

delay(1000);

lcd.print("1...");

delay(1000);

lcd.setCursor(0, 2);

lcd.print("Stopwatch started!");

j1=103;

j2=83;

j3=63;

j4=43;

j5=23;

j6=3;

showing();

do{

o = millis();

if (o - h > 1000){

h = o;

r[5]++;

if (r[5] == 10){

r[4]++;

r[5] = 0;

}

if (r[4] == 6 && r[5] == 0){

r[3]++;

r[4] = 0;

r[5] = 0;

}

if (r[3] == 10){

r[2]++;

r[3] = 0;

r[4] = 0;

r[5] = 0;

}

if (r[2] == 6 && r[3] == 0){

r[1]++;

r[2] = 0;

r[3] = 0;

r[4] = 0;

r[5] = 0;

}

if (r[1] == 10){

r[0]++;

r[1] = 0;

}

if (r[0] == 2 && r[1] == 4){

r[0] = 0;

r[1] = 0;

r[2] = 0;

r[3] = 0;

r[4] = 0;

r[5] = 0;

submenu = 0;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("One day has passed!");

lcd.setCursor(0, 1);

lcd.print("Ending Stopwatch");

delay(100);

lcd.print(".");

delay(100);

lcd.print(".");

delay(100);

lcd.print(".");

delay(100);

}

if (r[0] != r[6]){

r[6] = r[0];

//Hours d1

if (r[0] == 0){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 103;

//0

}

if (r[0] == 1){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 104;

//1

}

if (r[0] == 2){

for (z1 = 100; z1 < 120; z1++){

leds[z1] = CRGB(0, 0, 0);

FastLED.show();

}

j1 = 102;

//2

}

}

if (r[1] != r[7]){

r[7] = r[1];

//Hours d2

if (r[0] == 0){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 83;

//0

}

if (r[1] == 1){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 84;

//1

}

if (r[1] == 2){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 82;

//2

}

if (r[1] == 3){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 80;

//3

}

if (r[1] == 4){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 88;

//4

}

if (r[1] == 5){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 86;

//5

}

if (r[1] == 6){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 85;

//6

}

if (r[1] == 7){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 87;

//7

}

if (r[1] == 8){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 89;

//8

}

if (r[1] == 9){

for (z2 = 80; z2 < 100; z2++){

leds[z2] = CRGB(0, 0, 0);

FastLED.show();

}

j2 = 81;

//9

}

}

if (r[2] != r[2]){

r[8] = r[2];

//Minutes d1

if (r[2] == 0){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 63;

//0

}

if (r[2] == 1){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 64;

//1

}

if (r[2] == 2){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 62;

//2

}

if (r[2] == 3){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 60;

//3

}

if (r[2] == 4){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 68;

//4

}

if (r[2] == 5){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 66;

//5

}

if (r[2] == 6){

for (z3 = 60; z3 < 80; z3++){

leds[z3] = CRGB(0, 0, 0);

FastLED.show();

}

j3 = 65;

//6

}

}

if (r[3] != r[9]){

r[9] = r[3];

//Minutes d2

if (r[3] == 0){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 43;

//0

}

if (r[3] == 1){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 44;

//1

}

if (r[3] == 2){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 42;

//2

}

if (r[3] == 3){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 40;

//3

}

if (r[3] == 4){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 48;

//4

}

if (r[3] == 5){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 46;

//5

}

if (r[3] == 6){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 45;

//6

}

if (r[3] == 7){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 67;

//7

}

if (r[3] == 8){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 49;

//8

}

if (r[3] == 9){

for (z4 = 40; z4 < 60; z4++){

leds[z4] = CRGB(0, 0, 0);

FastLED.show();

}

j4 = 41;

//9

}

}

if (r[4] != r[10]){

r[10] = r[4];

//Seconds d1

if (r[4] == 0){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 23;

//0

}

if (r[4] == 1){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 24;

//1

}

if (r[4] == 2){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 22;

//2

}

if (r[4] == 3){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 20;

//3

}

if (r[4] == 4){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 28;

//4

}

if (r[4] == 5){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 26;

//5

}

if (r[4] == 6){

for (z5 = 20; z5 < 40; z5++){

leds[z5] = CRGB(0, 0, 0);

FastLED.show();

}

j5 = 25;

//6

}

}

if (r[5] != r[11]){

r[11] = r[5];

//Seconds d2

if (r[5] == 0){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 3;

//0

}

if (r[5] == 1){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 4;

//1

}

if (r[5] == 2){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 2;

//2

}

if (r[5] == 3){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 0;

//3

}

if (r[5] == 4){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 8;

//4

}

if (r[5] == 5){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 6;

//5

}

if (r[5] == 6){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 5;

//6

}

if (r[5] == 7){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 7;

//7

}

if (r[5] == 8){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 9;

//8

}

if (r[5] == 9){

for (z6 = 0; z6 < 20; z6++){

leds[z6] = CRGB(0, 0, 0);

FastLED.show();

}

j6 = 1;

//9

}

}

if (digitalRead(10) == 0){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Stopping timer!");

delay(600);

lcd.setCursor(0, 0);

lcd.print("Go back to menu?");

lcd.setCursor(1, 1);

lcd.print("YES");

lcd.setCursor(1, 2);

lcd.print("NO");

do{

if (0 < counter && counter < 5){

lcd.setCursor(0, 1);

lcd.write(1);

lcd.setCursor(0, 2);

lcd.print(" ");

if (digitalRead(10) == 0){

tone(po, 500, 100);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Just a second!");

delay(1000);

submenu = 0;

counter = 2;

cler = 0;

}

}

if (5 < counter && counter < 10){

o = millis();

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(0, 2);

lcd.write(1);

if (digitalRead(10) == 0){ lcd.clear(); lcd.setCursor(0, 0); lcd.print("OK");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Going back in:");

lcd.setCursor(0, 1);

lcd.print("10 seconds");

do{

o = millis();

if (o - h > 1000){

h = o;

e--;

if (e == 9){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Going back in:");

}

lcd.setCursor(0, 1);

lcd.print(e);

lcd.print(" seconds");

if (e == 0){

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Going back!");

submenu = 0;

counter = 2;

cler = 0;

return;

}

}

} while (submenu == 6);

}

}

} while (submenu == 6);

}

}

showing();

} while (submenu == 6);

}

}

}

void showing(){

//H[0]

leds[j1] = CRGB(r, g, b);

FastLED.show();

leds[j1 + 10] = CRGB(r, g, b);

FastLED.show();

//H[1]

leds[j2] = CRGB(r, g, b);

FastLED.show();

leds[j2 + 10] = CRGB(r, g, b);

FastLED.show();

//M[0]

leds[j3] = CRGB(r, g, b);

FastLED.show();

leds[j3 + 10] = CRGB(r, g, b);

FastLED.show();

//M[1]

leds[j4] = CRGB(r, g, b);

FastLED.show();

leds[j4 + 10] = CRGB(r, g, b);

FastLED.show();

//S[0]

leds[j5] = CRGB(r, g, b);

FastLED.show();

leds[j5 + 10] = CRGB(r, g, b);

FastLED.show();

//S[1]

leds[j6] = CRGB(r, g, b);

FastLED.show();

leds[j6 + 10] = CRGB(r, g, b);

FastLED.show();

}

ISR(PCINT0\_vect){

clk\_State = (PINB & B00000001);

dt\_State = (PINB & B00000010);

if (clk\_State != Last\_State){

if (dt\_State != clk\_State){

counter++;

}

else{

counter--;

}

}

Last\_State = clk\_State;

}