Upravljanje rasvjetom putem WiFi-a

Krušarovski, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij Elektrotehnika i informacijska tehnologija

Upravljanje rasvjetom putem WiFi-a

Diplomski rad

Vedran Krušarovski

Osijek, 2021.

Sadržaj

1. UVOD

Zadatak diplomskog rada je razvoj i izrada sustava za upravljanje rasvjetom putem WIFI-a temeljenog na mikrokontroleru.

Sustav koristi Atmel ATmega16 mikrokontroler koji u potpunosti kontrolira sustavom rasvjete, kako on nema mogućnost korištenja WIFI-a, s njime se spaja ESP8266-01 modul preko UART serijske komunikacije. Kako bi se korisniku omogućilo unaprijed postavljanje vremena paljenja i gašenja rasvjete, koristi se DS3231 senzor za vrijeme. Ovaj senzor na sebi ima ugrađenu pomoćnu bateriju koja mu omogućuje precizno praćenje vremena i kada sustav ostane bez napajanja.

Diplomski rad strukturiran je u četiri glavna poglavlja. Prvo poglavlje kratko opisuje hardver koji se koristi. Drugo poglavlje opisuje programe i programska okruženja. U trećem poglavlju opisan je programski kod koji je korišten za programiranje ATmega16 sa svim korištenim funkcijama. U ovom poglavlju također je opisana izrada mobilne aplikacije za upravljanje sustavom rasvjete. U zadnjem poglavlju opisano je korištenje sustava i mobilne aplikacije.

2. PREGLED PODRUČJA TEME

U današnje vrijeme postoji mnogo načina za kontroliranje rasvjete putem wifi-a. Neki od primjera su: Philips hue, Wyze. Takvi sustavi za kontroliranja rasvjete nude svoje mobilne aplikacije i često se integriraju sa sustavima kao što su Google Assistant, Amazon Aleksa, Apple HomeKit. To im omogućuje dodatne funkcionalnosti kao što je kontroliranje rasvjete putem glasa. Ovakva rješenja znaju biti poprilično skupa jer zahtijevaju kupnju posebnih „pametnih“ žarulja koje su skuplje od običnih žarulja. Ukoliko je potrebno kontroliranje više žarulja, ovakva rješenja brzo postanu skupa. Također, kada žarulja prestane raditi potrebno je ponovo kupiti „pametne“ žarulje, pa održavanje postane puno skuplje. Neki od ovakvih načina upravljanjem rasvjetom putem wifi-a često zahtijevaju dodatni uređaj (hub) na koji se žarulje moraju spojiti, što predstavlja još jednu dodatnu investiciju. U ovom diplomskom radu napravljen je sustav koji ne zahtjeva kupnju posebnih „pametnih“ žarulja već omogućuje korištenje „obične“ žarulje koje su znatno jeftinije.

Philips hue [1]

Načini upravljanja rasvjetom:

- Govorom preko Amazon Alexa, Google Assistant, HomeKit
- Preko mobilne aplikacije
- Ručno

Zahtjeva korištenje hub-a.

Koristi „pametne“ žarulje.

Wyze [2]

Načini upravljanja rasvjetom:

- Govorom preko Amazon Alexa, Google Assistant
- Preko mobilne aplikacije
- Ručno

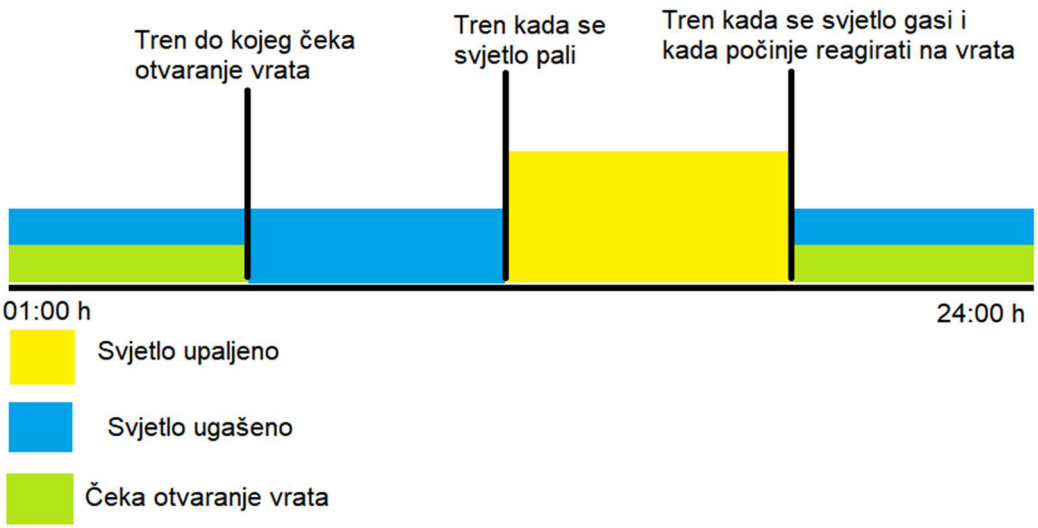
Ne zahtjeva korištenje hub-a.

Koristi „pametne“ žarulje.

3. KORIŠTENI HARDVER

3.1 Izvedba diplomskog rada

Sustav ima 3 intervala rada, interval u kojem pali rasvjetu, interval u kojem je rasvjeta ugašena no pali se ukoliko se ulazna vrata otvore i interval u kojem je sustav ugašen. Svaki od intervala korisnik može podesiti. Kada sustav uđe u interval za paljenje rasvjete, ovisno o satu i minutama, sustav se pali. Kada sustav dođe do kraja tog intervala rasvjeta će se ugasiti i sustav će ući u novi interval. U tom intervalu sustav će držati svjetla ugašenim sve dok se ulazna vrata ne otvore. Kad su se vrata otvorila, sustav će upaliti rasvjetu, nakon zatvaranja vrata pokreće se odbrojavanje, koje korisnik može podesiti od 0 do 59 minuta. Kada se postigne željeno vrijeme, sustav će ugasiti rasvjetu. Sustav se zadržava u tome intervalu dok se ne postigne određeno vrijeme. Sljedeći interval će rasvjetu držati ugašenom i neće reagirati na otvaranje / zatvaranje vrata. Slika 3.1. prikazuje intervale u kojima sustav radi.

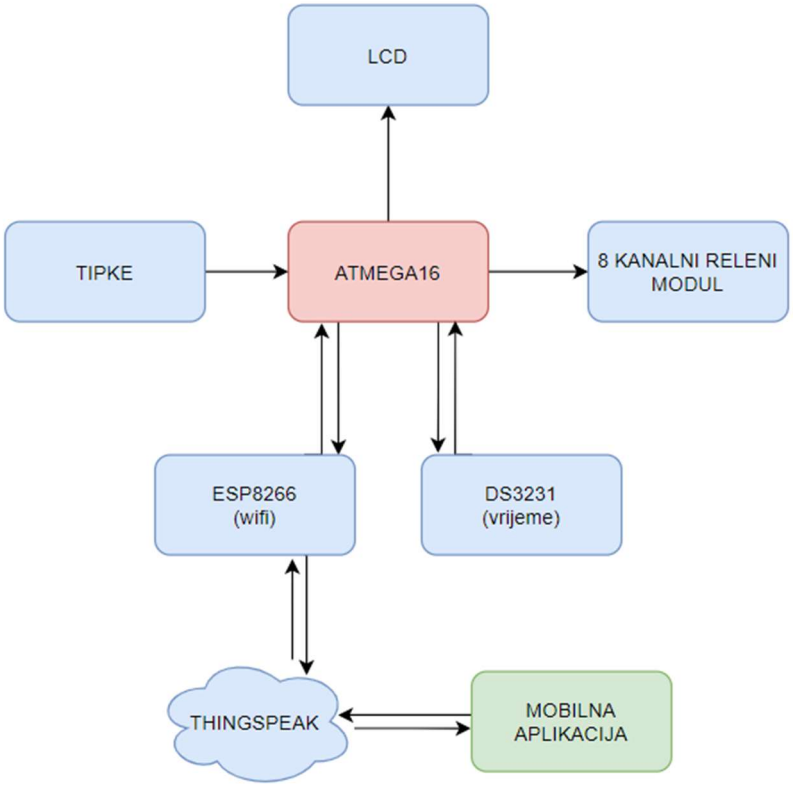


Slika 3.1. Prikaz svih intervala sustava

Mikrokontroler ATmega16 napaja se preko računala putem USB (*engl. Universal Serial Bus*) priključka korištenjem USB AVR (*engl. Automatic Voltage Regulator*) programatora. Programiranje mikrokontrolera također se vrši korištenjem USB AVR programatora.

Na ATmega16 je spojen LCD zaslon 16x2 preko porta A, preko porta B spojene su tipke za upravljanje sustava za rasvjetu i prekidač koji reagira na otvaranje vrata. Sustav ima 6 tipki: naprijed, nazad, plus, minus, uredi i postavi. Na portu C ATmega16, spojena su dva releja i DS3231 senzor. DS3231 je spojen na pinovima 0 i 1 te kao takav koristi I^2C komunikacijski protokol. Na portu D spojeno je 6 releja i ESP8266 modul. Modul ESP8266 spojen je na port D preko pinova 0 i 1 i koristi UART komunikaciju.

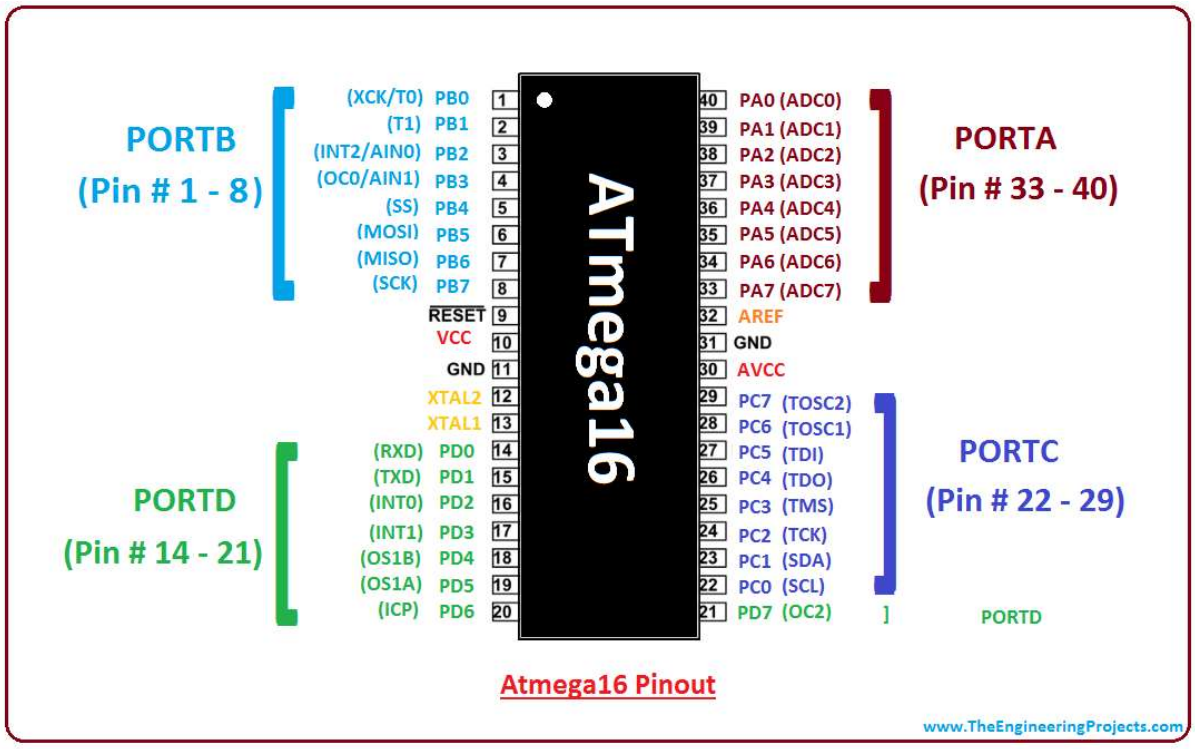
Za kontrolu sustava preko interneta napravljena je mobilna aplikacija za Android operativni sustav. Nadziranje sustava preko interneta, može se preko mobilne aplikacije ili preko stranice ThingSpeak.



Slika 3.2. Blokovski prikaz sustava

3.2 Atmel ATmega16

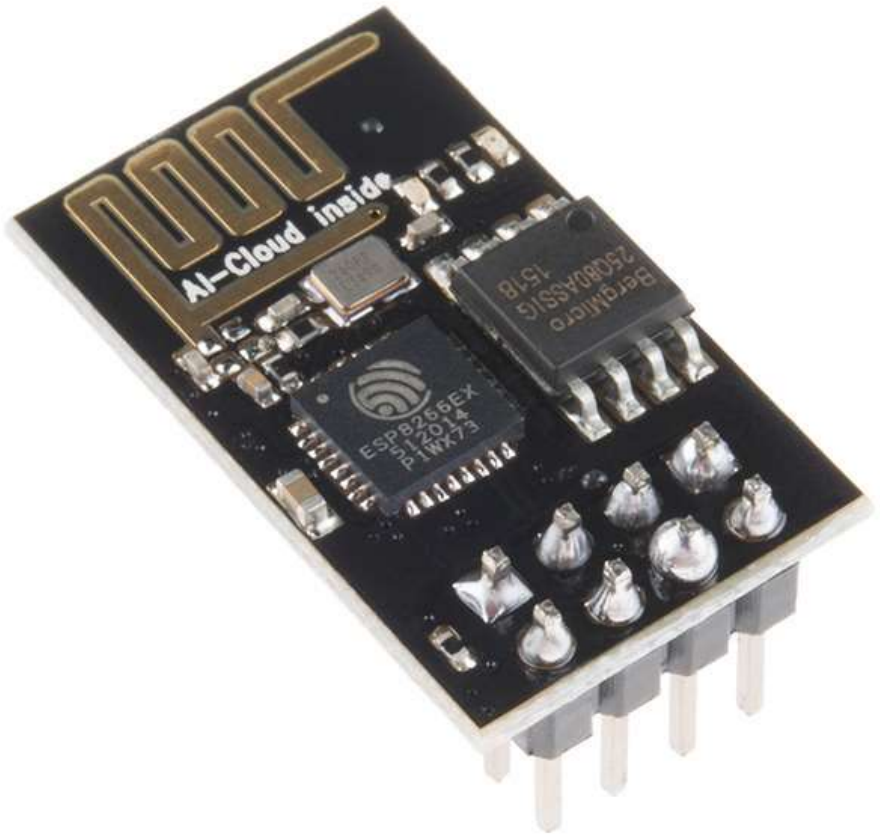
Za ovaj sustav izabran je Atmel ATmega16 mikrokontroler. Ovaj mikrokontroler je 8-bitni AVR mikrokontroler baziran na poboljšanoj RISC (engl. *Reduced instruction set computer*) arhitekturi. Ima 16 kB *flash* memorije, 512 B EEPROM-a (engl. *Programmable Read-Only Memory*). Radi na naponu od 4,5 do 5,5 V. Brzina ovog mikrokontrolera je 16MHz [3].



Slika 3.3. ATmega16 sa naznačenim pinovima [4]

3.3 Wi-Fi Modul ESP8266-01

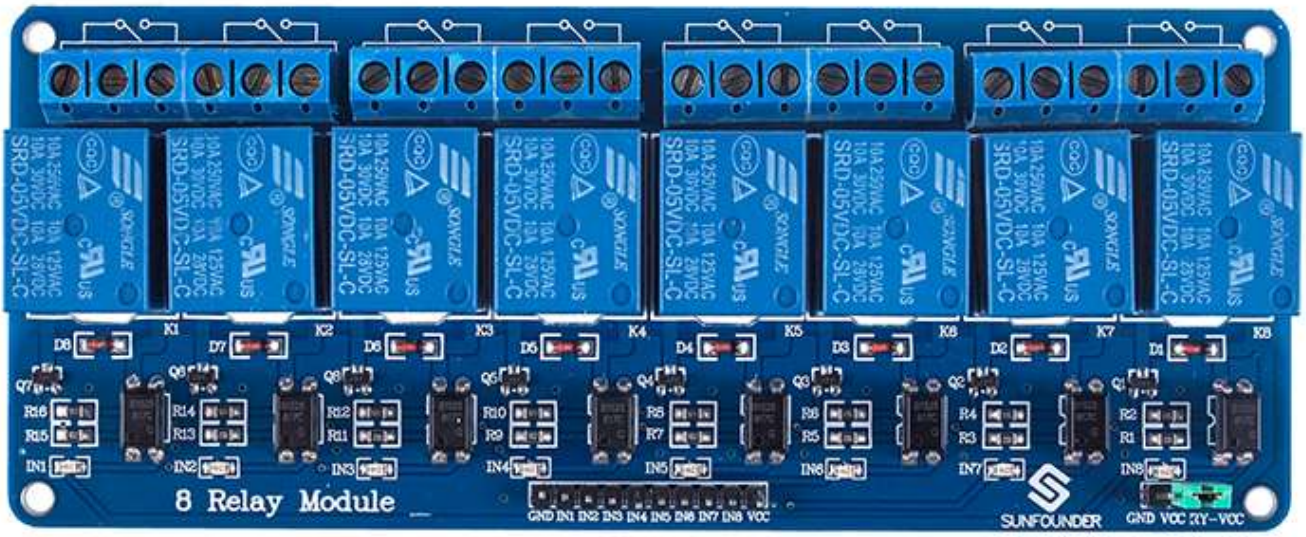
ESP8266-01 je Wi-Fi modul koji omogućuje mikrokontrolerima pristup Wi-Fi mreži. Ovaj modul koristi 32-bitni Tensilica L106 RISC procesor što omogućuje nisku potrošnju energije. Modul na sebi ima ugrađenu trakastu antenu[5]. Za komunikaciju između mikrokontrolera i ESP8266 modula koristi se UART protokol. Mikrokontroler izvršava komunikaciju s modulom slanjem AT naredbe. Postoji 5 osnovnih AT naredbi, 11 naredbi na Wi-Fi sloju i 11 naredbi na TCP/IP sloju. Naredbe mogu imati do 4 varijante ovisno o zadanom parametru na kraju komande. Varijante su Test, Query, Set, Execute [6]



Slika 3.4. ESP8266-01 modul [7]

3.4 Relej

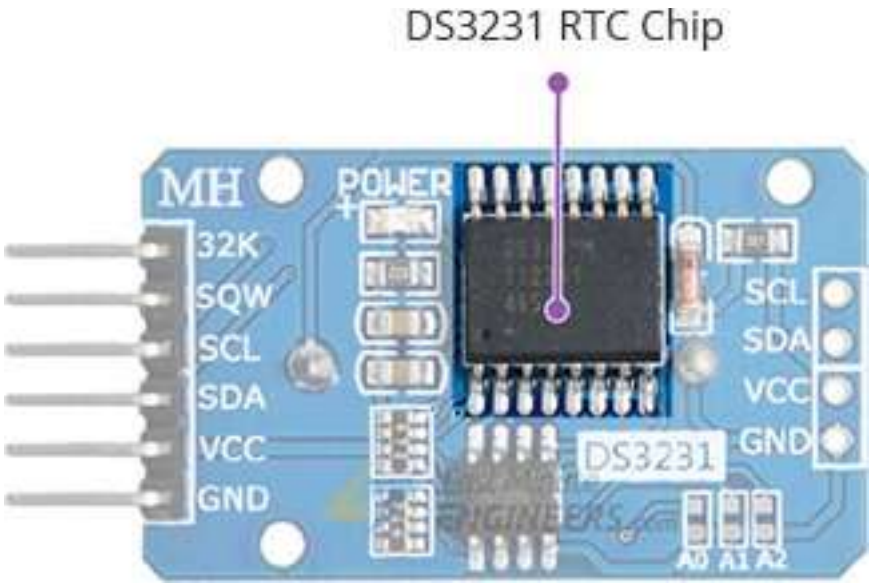
Relej je elektromagnetski prekidač koji niskim vrijednostima napona i struje upravlja znatno većim vrijednostima napona i struje. U ovom diplomskom radu koristi se modul sa 8 releja koji služi za paljenje i gašenje rasvjete na osnovu izlaznog signala s mikrokontrolera ATmega16. Za ovaj diplomski rad izabran je 8-kanalni relejni modul s naponom upravljanja od 5 V.



Slika 3.5. 8-kanalni relejni modul [8]

3.5 Senzor za vrijeme DS3231

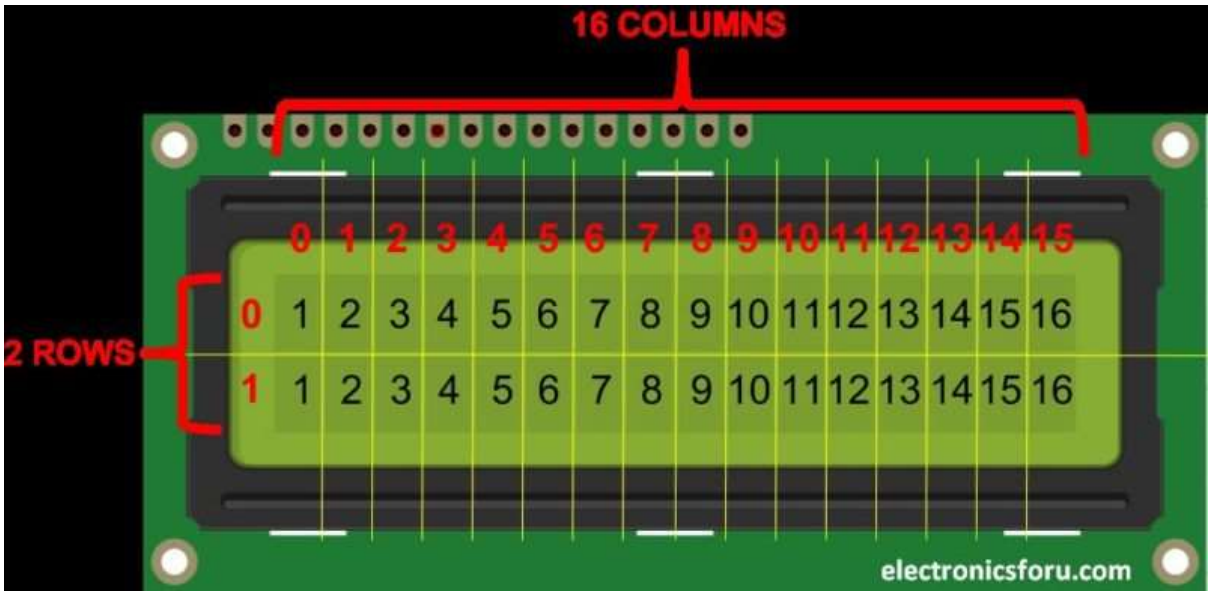
DS3231 je niskobudžetni modul za praćenje vremena. Na sebi ima integrirani temperaturno kompenziran kvarcni kristal što mu omogućuje precizno praćenje vremena pri raznim temperaturama. Preciznost mu iznosi ±2ppm na temperaturama od 0°C do +40°C. Preciznost pri temperaturi od -40°C do +85°C mu iznosi ±3.5ppm. Preciznost temperaturnog senzora iznosi ±3°C . DS3231 senzor ima i bateriju koja mu omogućuje praćenje vremena i kada izgubi primarno napajanje [9]. DS3231 senzor je s mikrokontrolerom spojen preko I2C protokola. Slika 3.4 prikazuje DS3231 modul s označenim DS3231 RTC čipom.



Slika 3.6. DS3231 modul sa označenim DS3231 RTC čipom [10]

3.6 LCD zaslon 16x2

Za prikaz informacija o stanju rasvjete i uspostave Wi-Fi konekcije koristi se LCD (eng. Liquid Crystal Display) zaslon 16x2. Ovaj zaslon može prikazati maksimalno 32 simbola odnosno 16 simbola po jednom redu. Svaki simbol je prikazan pomoću 5x7 *pixel* matrice [11]. Slika 2.5. prikazuje grafički prikaz 16x2 LCD zaslona s prikazanim mjestima svih mogućih simbola.



Slika 3.7. Grafički prikaz 16x2 LCD zaslona [11]

3.7 AC/DC pretvarač HLK-PM05

Kako relejni modul korišten u ovom radu zahtijeva istosmjerni napon od 5 V, koristi se AC/DC pretvarač HLK-PM05. HLK-PM05 na ulazu može primiti izmjenični napon od 90 do 264 V a na izlazu daje istosmjerni napon od 5 V (±0,1 V) i struju do 1 A. Efikasnost ovog AC/DC pretvarača pri ulazu od 220 V je minimum 70% pri maksimalnom opterećenju [12].

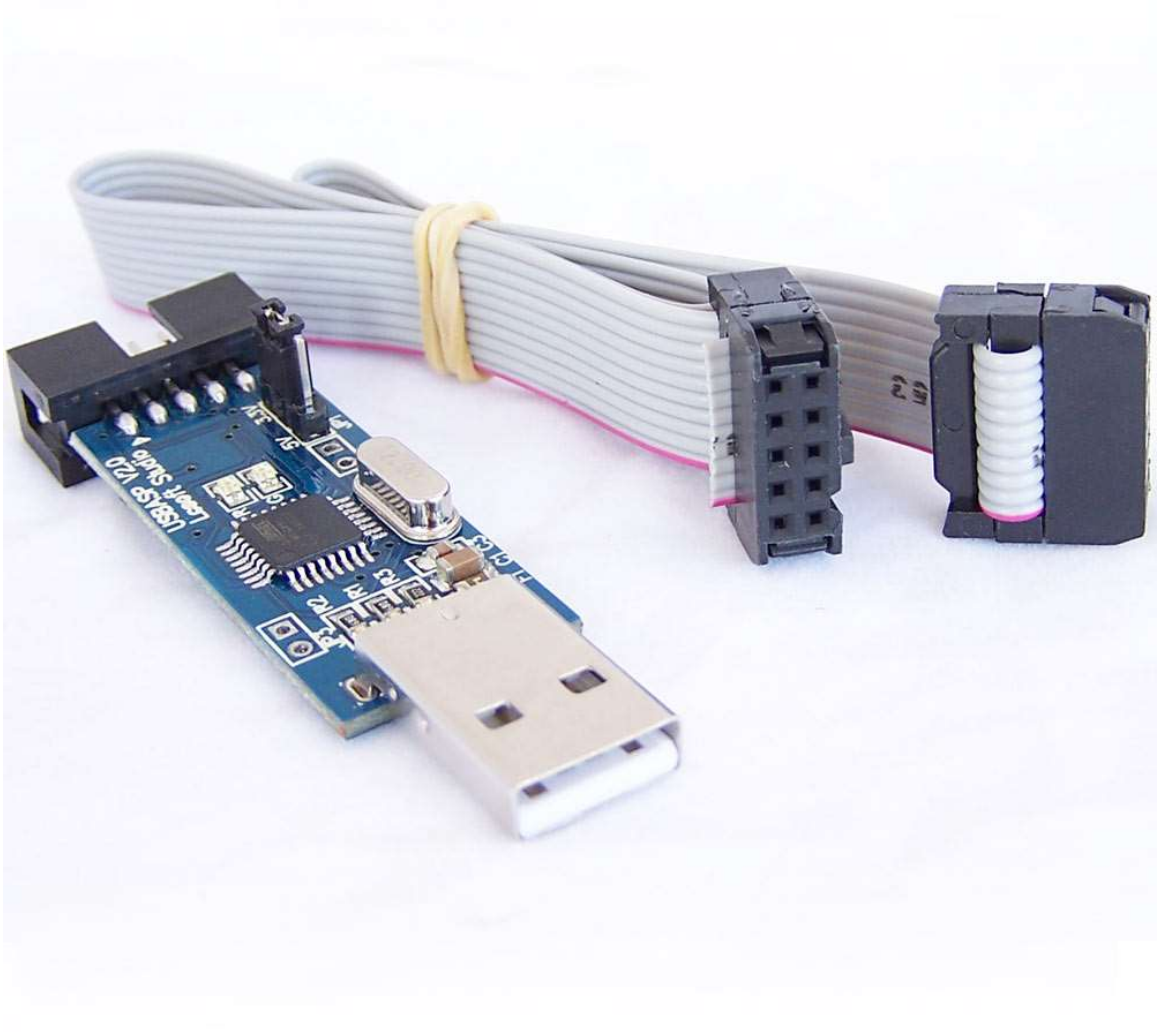
Slika 3.8. Hi-Link HLK-PM05 AC/DC pretvarač [12]

3.8 USBasp programator

USBasp je USB programator za Atmel AVR kontrolere. Sastoji od ATMega88 ili ATMega8 i nekoliko pasivnih komponenata [13]. Korištenjem ovog programatora omogućujemo komunikaciju između ATmega16 i kompjutera te tako možemo isprogramirati mikrokontroler.

Značajke [13]

- Radi na više platformi, Linux, Mac OS X i Windows.
- Brzina programiranja je do 5kB/s
- Nisu potrebni posebni kontroleri ili smd komponente.



Slika 3.9. USBasp programator [14]

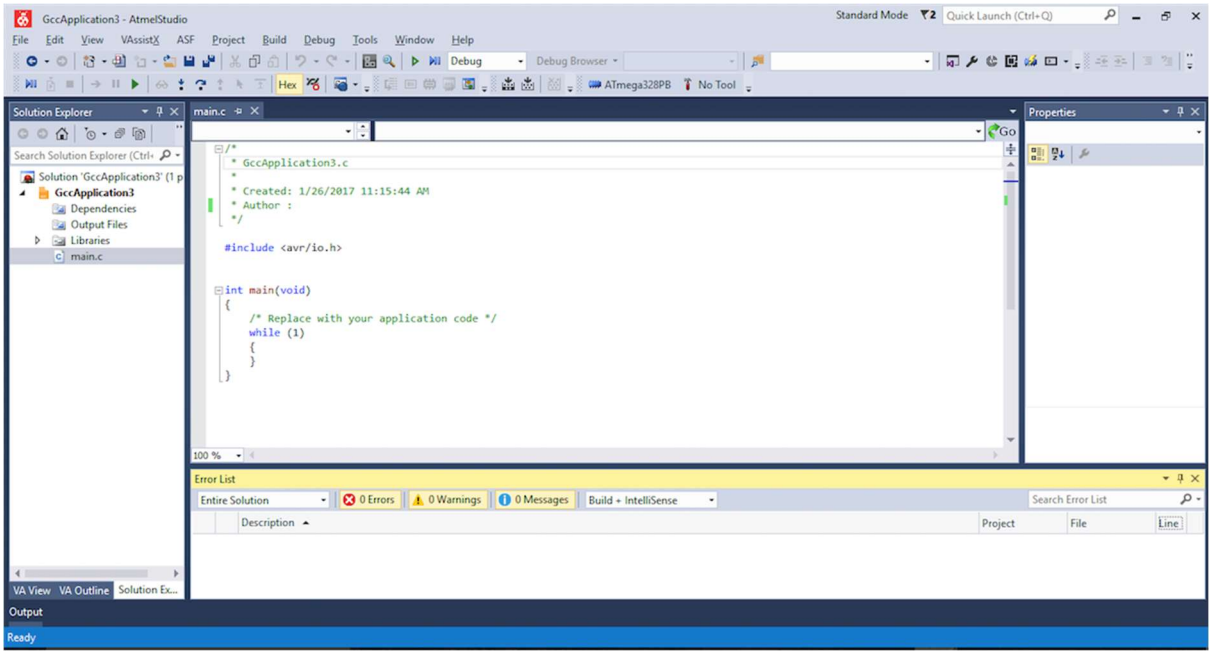
4. KORIŠTENI PROGRAMI I PROGRAMSKA OKRUŽENJA

4.1 Atmel studio 7

Atmel Studio 7 je integrirana razvojna platforma IDP (*engl. integrated development platform*) za razvoj i ispravljanje pogrešaka u aplikacijama temeljenim na SMART ARM i AVR mikrokontrolerima napisane u programskom jeziku C [15].

Glavne značajke [15]

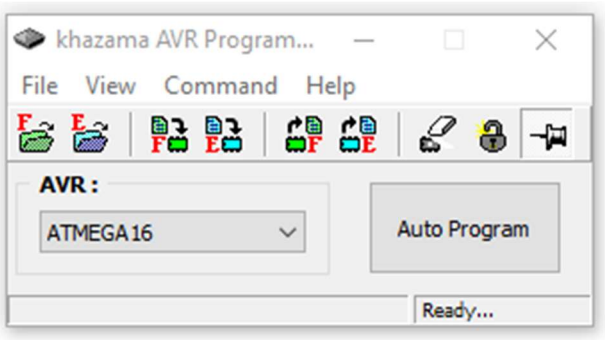
- Podrška za 300+ AVR i SMART ARM baziranih uređaja.
- Veliki broj biblioteka i drivera, preko 1600 primjera projektnih zadataka.
- Pisanje i otklanjanje grešaka u programskom jeziku C/C++.



Slika 4.1. Atmel studio 7 [15]

4.2 Khazama AVR programator

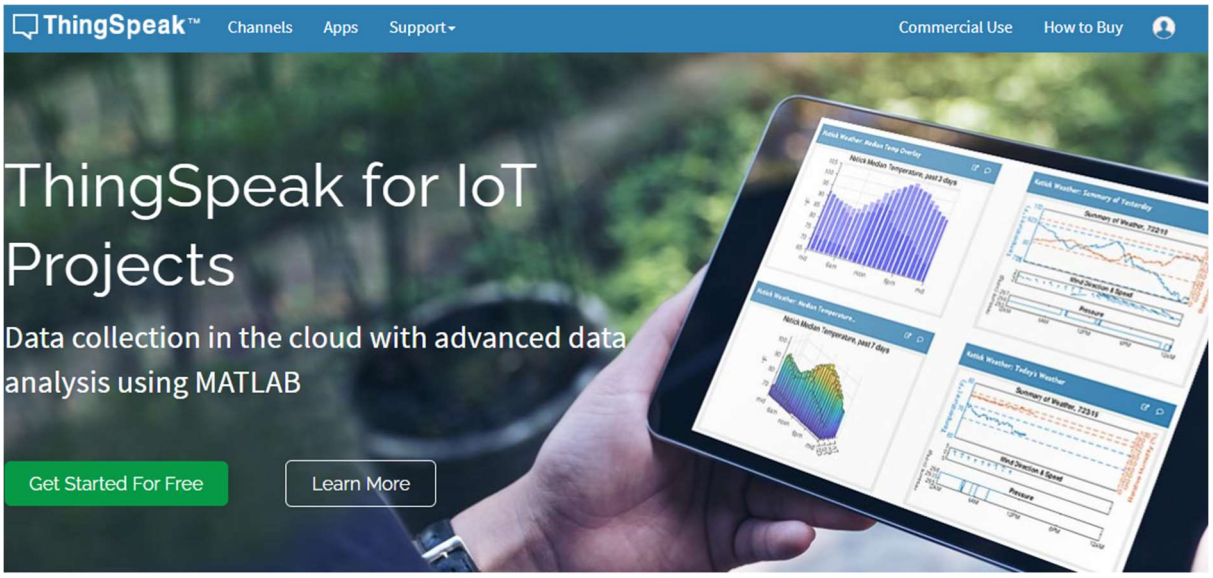
Khazama AVR programator služi za programiranje AVR mikrokontrolera. Neke od funkcija koje pruža ovaj program su: čitanje „potpisa“ čipa, stavljanje željene datoteke u *buffer* programatora, čitanje i postavljanje *Fuse* i *Lock* bitova. Kako bi se isprogramirao mikrokontroler potrebno je prvo odabrati mikrokontroler koji se koristi, kako se u ovom diplomskom radu koristi ATmega16, potrebno je njega odabrati. Programski kod koji želimo staviti na mikrokontroler, potrebno je staviti u .hex datoteci. Ta datoteka se stavlja u *buffer* (File > Load FLASH file to BUFFER) i zatim se stisne tipka Auto Program. Na kraju programiranja dobit će se poruka o uspješnom programiranju.



Slika 4.2. Khazama AVR programator

4.3 ThingSpeak

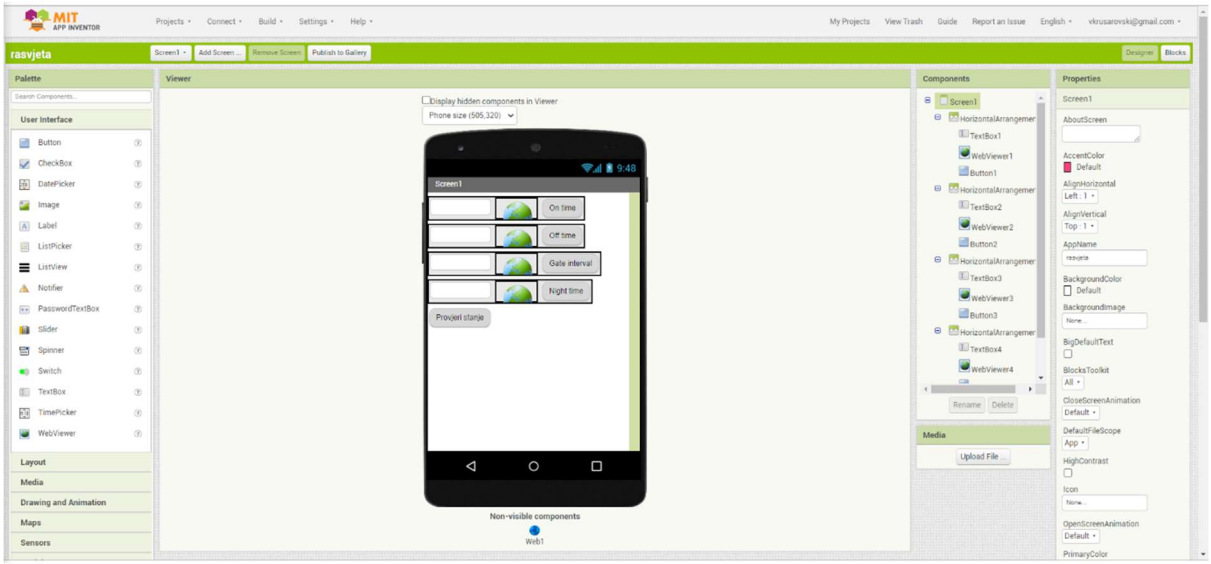
ThingSpeak je IoT (*engl. Internet of things*) analitička platforma koja omogućuje prikupljanje, vizualizaciju i analizu podataka uživo u oblaku. Omogućuje primanje podataka sa raznih uređaja i stvaranje trenutne vizualizacije podataka i slanje upozorenja preko web usluga kao što su Twitter i Twilio. ThingSpeak omogućuje inženjerima i znanstvenicima izradu prototipa i izgradnju IoT sustava bez postavljanja poslužitelja ili razvoja web softvera [16].



Slika 4.3. ThingSpeak web platforma [16]

4.4 Mit App Inventor

MIT App Inventor je intuitivno, vizualno programsko okruženje koje omogućuje izradu potpuno funkcionalnih aplikacija za Android i iOS pametne telefone i tablete. Mit App Inventor temeljen je na blokovima što olakšava stvaranje složenih aplikacija s visokim učinkom u znatno kraćem vremenu od tradicionalnih programskih okruženja [17].



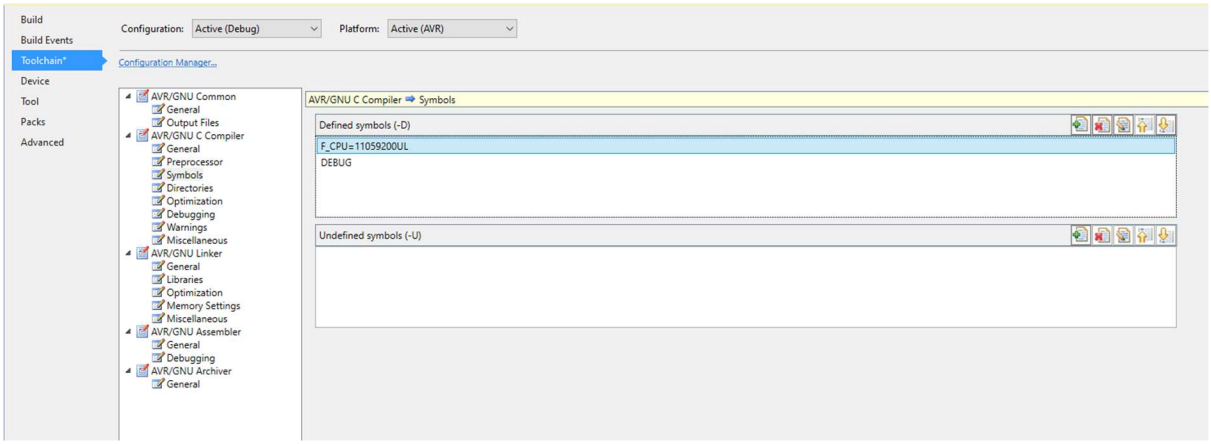
Slika 4.4. MIT App Inventor

5. IZRADA PROGRAMSKOG KODA

5.1 Početne postavke

5.1.1 Atmel Studio 7

Kod pravljenja novog projekta u programu Atmel Studio 7 prvo se mora odabrati mikrokontroler za koji pišemo programski kod. Kako se u ovom diplomskom radu koristi ATmega16, potrebno je njega odabrati. Nakon toga potrebno je definirati frekvenciju mikrokontrolera. Ukoliko projekt koristi više datoteka potrebno je frekvenciju definirati u svakoj datoteci ili se može zapisati unutar Atmel Studia 7 pod *Properties-Toolchain-AVR/GNU C Compiler-Symbols* kao sto je prikazano na slici 5.1.



Slika 5.1. Definiranje frekvencije za projekt od više datoteki

Frekvencija mikrokontrolera u ovom diplomskom radu je postavljenja kao `#define F_CPU 11059200UL`.

Kada se unutar Atmel Studia 7 postavila vrijednost frekvencije potrebno je namjestiti Lock bitove na ATmega16 kako bi koristio vanjski oscilator. To se napravi unutar programa Khazama AVR programatora *Command-Fuses and Lock Bits*.

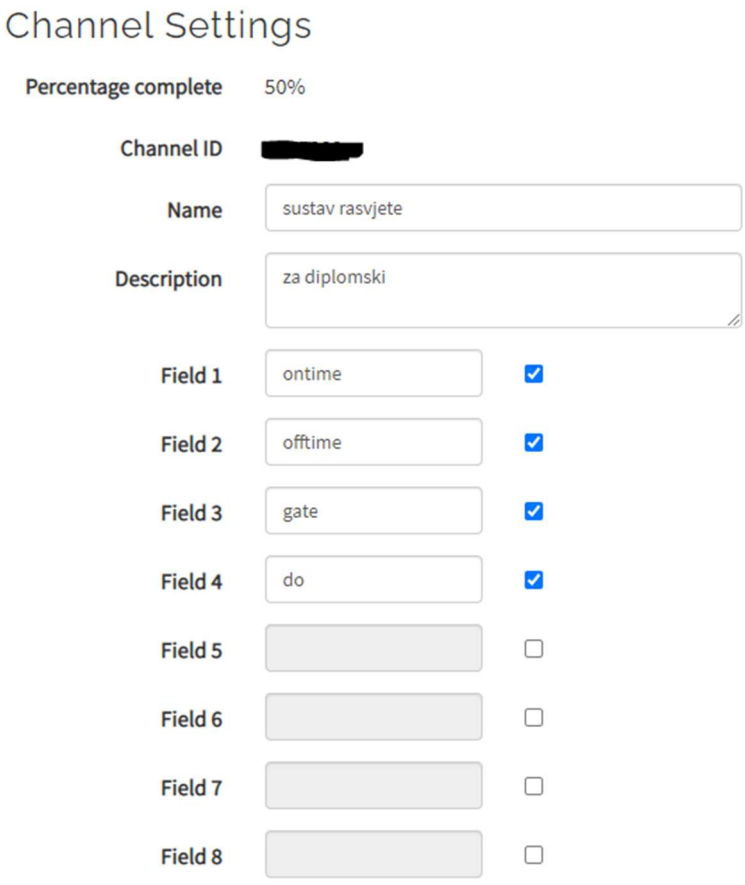
Kada je uspješno postavljenja frekvencija potrebno je uključiti sve datoteke koje se koriste kao što je prikazano na slici 5.2.

```
#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "I2C_Master_H_file.h"
#include "lcd.h"
#include <util/delay.h>
#include <avr/interrupt.h>
#include "Uart.h"
#include <stdlib.h>
```

Slika 5.2. prikaz svih korištenih datoteka

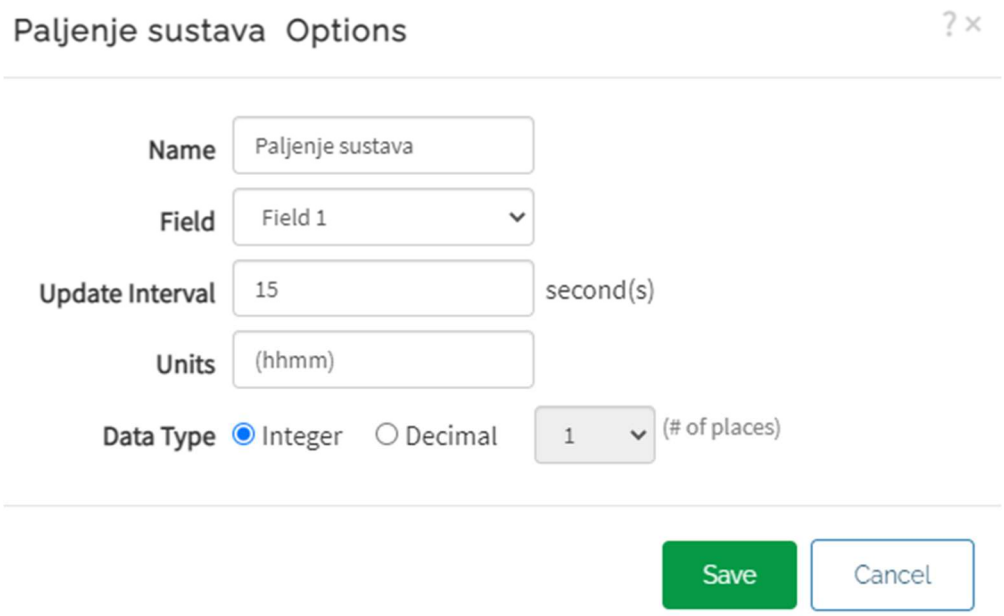
5.1.2 ThingSpeak

Kako bi omogućili spremanje podataka na internetu koristimo platformu ThingSpeak koja omogućuje spremanje i prikaz podataka. Prvo je potrebno napraviti račun na stranici, a zatim napraviti novi kanal, dati mu ime i postaviti koliko će se polja koristiti. Na slici 5.3 se vidi kako treba izgledati kanal za ovaj diplomski rad.

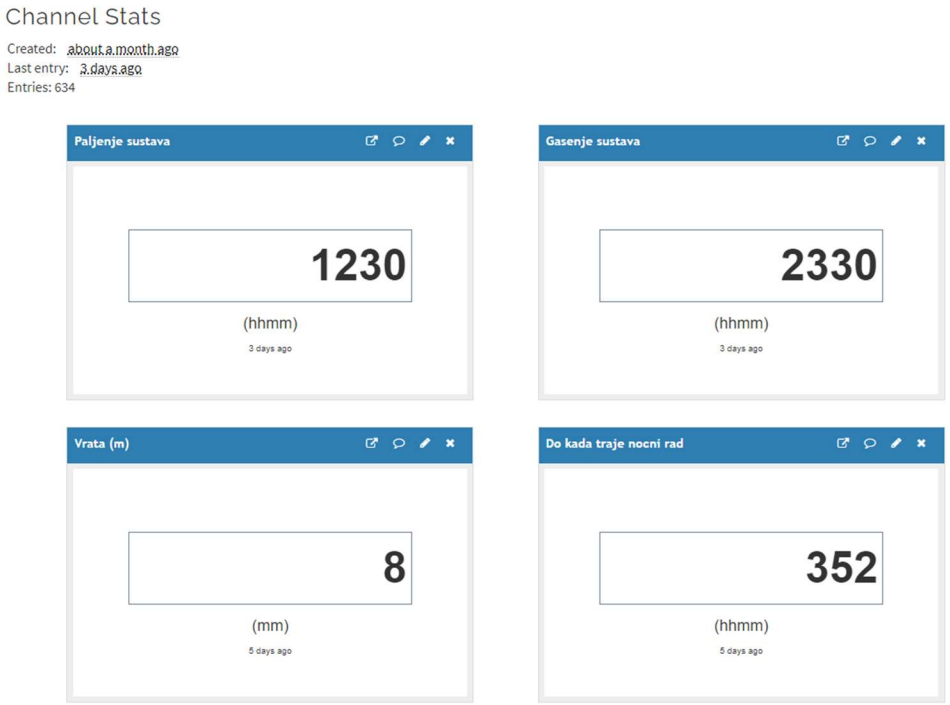


Slika 5.3. Konfiguracija ThingSpeak kanala

Nakon toga na stranici se mogu postaviti prozori za bolji pregled podataka na sljedeći način, Add Widgets > Numeric Display, zatim je potrebno ispuniti što će se pokazivati unutar tog prozora, kao što je prikazano na slici 5.4. Kada se podese svi prozori s odgovarajućim poljima izgled stranice ThingSpeak će izgledati kao na slici 5.5.



Slika 5.4. Konfiguracija jednog prozora



Slika 5.5. Izgled stranice ThingSpeak nakon podešavanja prozora

5.2 Opis programskog koda

U ovom diplomskom radu koristio sam gotove „biblioteke“ za I2C [18] i lcd [19] .

Pri prvom pokretanju koda, potrebno je navesti ime WIFI mreže i lozinku od mreže na koju će se sustav spojiti. Ti podaci se upisuju kod definiranja SSID i PASSWORD. S gore navedenim djelom omogućeno je spajanje na Internet preko WIFI mreže. Kako ovaj diplomski rad koristi platformu ThingSpeak, definirani dijelovi za DOMAIN i PORT trebaju biti "api.thingspeak.com" i "80". Definirani dio API_WRITE_KEY i CHANNEL_ID potrebno je promijeniti u odgovarajuće vrijednosti pojedinačnog korisnika i njegovog kanala. Ti podaci se pročitaju sa platforme ThingSpeak. Slika 5.6. prikazuje dijelove koje potrebno definirati za pristup WIFI mreži i ThingSpeak platformi.

```
#define SSID              "SSID"
#define PASSWORD          "password"
#define DOMAIN            "api.thingspeak.com"
#define PORT              "80"
#define API_WRITE_KEY     "BLPU**********"
#define CHANNEL_ID        "14*****"
```

Slika 5.6. Prikaz gdje je potrebno navesti podatke stranice i wifi mreže

Također, pri prvom programiranju potrebno je *otkomentirati* i postaviti vrijednosti za sat i kalendar, slika 5.7. Ti podaci se upisuju u RTC. Kako DS3231 ima vlastitu bateriju i u slučaju gubitka struje RTC DS3231 će nastaviti vjerno bilježiti vrijeme. Iz tog razloga ovaj dio koda potrebno je samo jednom izvršiti.

```
//RTC_Clock_Write(0x20, 0x01, 0x00, hour_24); //Zapisuje sat, minute, sekunde u tocno ovom formatu
//RTC_Calendar_Write(0x1, 0x30, 0x8, 0x21); // Zapisuje redni broj dana u tjednu, dan u mjesecu, mjesec i godinu
```

Slika 5.7. Prikaz šta se treba otkomentirati za zapis vremena i datuma

Kako senzor DS3231 koristi I2C komunikaciju koristimo „biblioteku“ I2C_Master_H_file koja je zadužena za inicijalizaciju I2C na mikrokontroleru i komunikaciju između RTC-a DS3231 i ATmega16.

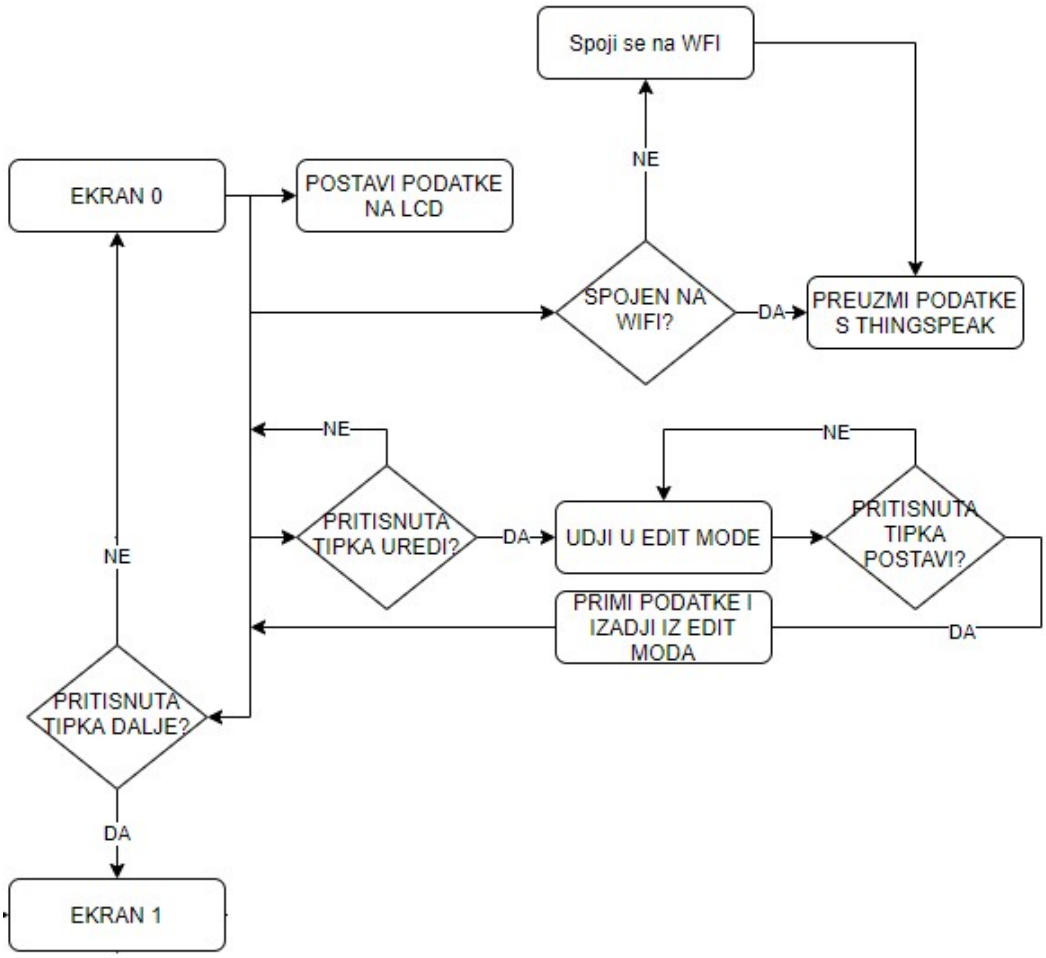
Za kontrolu lcd-a 16x2 koristi se „biblioteku“ lcd u kojoj su definirane funkcije za inicijalizaciju lcd-a, postavljanje kursora na određeno mjesto na lcd-u, ispis znakova na lcd-u, brisanje svih simbola s ekrana.

Kako esp8266 koristi komunikaciju UART koristi se i datoteka Uart.c u kojoj se izvršava inicijalizacija uarta, i ima funkcije za slanje i primanje jednog znaka i slanje stringa.

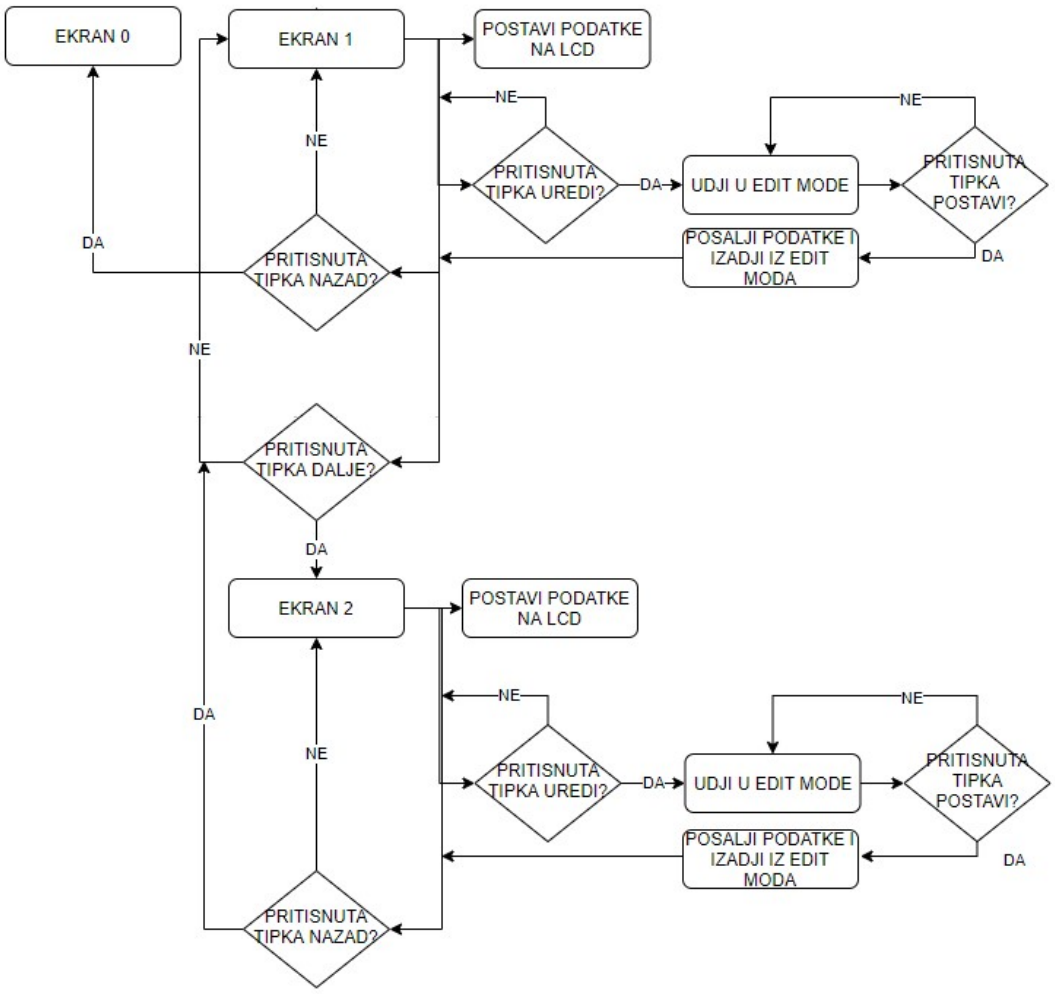
U main.c su prvo je definirana frekvencija od 11059200 hz, zatim su definirani pinovi koji se koriste za tipke na portu B. Također definirane su i adrese za pisanje i čitanje sa DS3231 i inicijalizirane su sve globalne varijable koje se koriste u projektu. Nakon toga dolaze funkcije korištene u ovom projektu.

U funkciji main nakon postavljanja portova za ulaz i izlaz, inicijalizira se I^2C, UART, koji radi pri brzini od 115200 bauda, i inicijalizira se lcd. Nakon toga šaljemo AT komandu sa ATmega16 na esp8266 kako bi saznali dali je esp8266 spojen na wifi mrežu, ukoliko nije pokušati će se spojiti.

S time je izvršen dio programskog koda koji se vrti samo jednom pri pokretanju sustava, i ulazi u petlju koja se stalno ponavlja. U toj petlji pomoću Switch case petlje i screenPicker varijable pratimo na kojem zaslonu se moramo nalaziti. Na početnom zaslonu (ekran 0) prati se, dali je tipka “naprijed” ili tipka ”uredi” pritisnuta. Ako je pritisnuta tipka “uredi” ulazimo u način rada za izmjenu podataka. Na ekranu 0 možemo promijeniti vrijednosti sata i minuta. Kako bi se izašlo iz tog načina rada potrebno je pritisnuti tipku “postavi” kada će se promijenjeno vrijeme poslati na RTC. Sustav će se svake minute spojiti na ThingSpeak stranicu i primat će podatke s nje, kako bi napravio ažuriranje lokalnih postavki. Sustav će to napraviti samo ako je esp8266 spojen na wifi mrežu. Dok sustav radi ažuriranje lokalnih postavki na ekranu će se pojaviti simbol *, kao sto je prikazano na slici 6.2. Dok god je prikazan taj simbol sustav će pauzirati sa svim ostalim radnjama dok se ažuriraju lokalne postavke. Dijagram toka za početni ekran prikazan je na slici 5.8, dok dijagrami toka za ekran 1 i ekran 2, su prikazani na slici 5.9.



Slika 5.8. Dijagram toka za ekran 0



Slika 5.9. Dijagram toka za ekran 1 i 2

Ako je pritisnuta tipka naprijed na ekranu 0 povećavamo vrijednost varijable screenPicker za 1, obrišemo sve što se nalazi na ekranu te pričekamo 200ms kako bi osigurali da pritisak tipke se registrira samo jednom. Isto to će se desiti i na ostalim ekranima ali ne i na zadnjem. Na zadnjem sustav reagira na tipku "nazad". Na ekranima 0, 1 i 2 sustav će reagirati na pritisak tipke "uredi", kada će ući u način rada za ispravljanje podataka. Pomoću još jedne Switch case petlje i "editMenu" varijable određujemo koju vrijednost trenutno možemo promijeniti, koristeći tipke "naprijed" i "nazad", povećava se varijabla „editMenu" za 1 te se ulazi u sljedeći slučaj Switch case petlje. To omogućuje biranje kojeg podatka na ekranu želimo promijeniti. Odabranu vrijednost možemo povećati ili smanjiti pritiskom na tipke „plus" ili „minus". Kada se sustav nalazi u modu za izmjenu podataka na ekranu će se pojaviti znak < ili > (slika 6.3) koji će nam pokazati koji podatak trenutno možemo promijeniti. Na ekranima 1 i

2, kada se izlazi iz načina rada za ispravljanje, na ekranu će se pojaviti natpis „UPDATING“ što pokazuje da se podaci šalju na ThingSpeak platformu.

Nakon prve Switch case petlje, dolazimo do dijela koda koji je odgovoran za paljenje i gašenje rasvjete. U prvom dijelu se prati da li je zadovoljen uvjet za paljenje rasvjete i gašenje rasvjete. Nakon tog dijela, dolazi dio koji omogućuje rad sustava u intervalu kada će sustav reagirati na otvaranje vrata. Ovaj dio koda je prikazan na slici 5.10.

```
        if (gateFlag == false)                              // ovaj dio koda ce se izvrsavat samo ako su vrata zatvorena (nisu bila otvorena u odredjenom intervalu)
        {
            if ( (psat > hourONtime) && (psat < hourOFFtime) )  // ovaj dio gleda kada su sati > <
                light = true;
            else if (psat == hourONtime && pmin >= minuteONtime) // ovaj dio gleda za tocno odredeni sat i minute ON TIME
                light = true;
            else if (psat == hourOFFtime && pmin < minuteOFFtime)  // ovaj dio gleda za tocno odredeni sat i minute OFF TIME
                light = true;
            else
                light = false;
        }
//******** gate stuff *********************************************
        if ((psat > hourOFFtime) && (psat < gateOFFtimeH))      // ovo ce se izvrsavat od trenutka gasenja svijetla do gate off time
            gateEvent();
        else if ((psat == hourOFFtime) && (pmin >= minuteOFFtime))
            gateEvent();
        else if (psat == gateOFFtimeH && pmin <= gateOFFtimeM )
            gateEvent();
        else if (gateOFFtimeH < hourOFFtime)        // ako je vrijeme gasenja od vrata manje od OFF time
        {                                           // podjeli na dva intervala i provjeri u kojem je
            if (psat > hourOFFtime && psat <= 24)   // od gasenja do ponoci
                gateEvent();
            if (psat < gateOFFtimeH)            // od 01 do gasenja od vrata
                gateEvent();
        }

        if (gateFlag == true)       // ako su vrata otvorena/bila otvorena pocni odbrojavanje
            gateTimer();
        if (gateFlag == true)       // ako su vrata otvorena/bila otvorena (i unutar intervala od timera) upali sv
            light = true;

        if (light)          //upali svjetlo ovisno o zastavici za light
            turnON();
        else
            turnOFF();      //ugasi svjetlo ovisno o zastavici za light
```

Slika 5.10. Prikaz koda za paljenje i gašenje svjetla ovisno vremenu

5.3 Korištene funkcije

```
void RTC_Clock_Write(char _hour, char _minute, char _second, char AMPM)
```

Funkcija koja zapisuje vrijeme na DS3231. Vrijeme na DS3231 se nalazi na registru 0, točnije, sekunde se nalaze na registru 00, minute na 01 i sat na 02. Ova funkcija ne vraća nikakve podatke, no prima sljedeće vrijednosti:

char _hour - Vrijednost sata koja se upisuje u RTC

char _minute - Vrijednost minute koja se upisuje u RTC

char _second - Vrijednost sekunde koja se upisuje u RTC

char AMPM - za postavljanje sata u način AM PM.

```
void RTC_Clock_Write(char _hour, char _minute, char _second, char AMPM)
{
    _hour |= AMPM;
    I2C_Start(Device_Write_address);    // Zapocni I2C komunikaciju sa RTC
    I2C_Write(0);                       // Zapisi na 0 lokaciju, za vrijednosti sekunde
    I2C_Write(_second);                 // Zapisi na lokaciju 00 koja prima vrijednost sekunde
    I2C_Write(_minute);                 // Zapisi na lokaciju 01 koja prima vrijednost minute
    I2C_Write(_hour);                   // Zapisi na lokaciju 02 koja prima vrijednost sate
    I2C_Stop();                         // Zaustavi I2C komunikaciju
}
```

Slika 5.11. Funkcija RTC_Clock_Write

```
void RTC_Calendar_Write(char _day, char _date, char _month, char _year)
```

Funkcija koja zapisuje datum na DS3231. Datum na DS3231 se nalazi na registru 03, točnije, dan u tjednu (redni broj) se nalaze na registru 03, dan u mjesecu na 04, mjesec na 05 i godina na registru 06. Ova funkcija ne vraća nikakve podatke, no prima sljedeće vrijednosti:

char _day - Vrijednost dana u tjednu koja se upisuje u RTC

char _date - Vrijednost dana u mjesecu koja se upisuje u RTC

char _month- Vrijednost mjeseca koja se upisuje u RTC

char _year - Vrijednost godine koja se upisuje u RTC.

```
void RTC_Calendar_Write(char _day, char _date, char _month, char _year) /* function for calendar */
{
    I2C_Start(Device_Write_address);    // Zapocni I2C komunikaciju sa RTC
    I2C_Write(3);                       // Zapisi na 3 lokaciju, za vrijednosti dana
    I2C_Write(_day);                    // Zapisi na lokaciju 03 koja prima vrijednost dana, redni broj dana u tjednu
    I2C_Write(_date);                   // Zapisi na lokaciju 04 koja prima vrijednost datuma/dan
    I2C_Write(_month);                  // Zapisi na lokaciju 05 koja prima vrijednost datuma/mjesec
    I2C_Write(_year);                   // Zapisi na lokaciju 06 koja prima vrijednost datum/godina
    I2C_Stop();                         // Zaustavi I2C komunikaciju
}
```

Slika 5.12. Funkcija RTC_Calendar_Write

```
void rtc_get_temp()
```

Funkcija zadužena za dohvaćanje temperature sa DS3231, na n1 je spremljena vrijednost temperature, dok je na n2 spremljena decimalna vrijednost. Kako DS3231 na registru 0x13 sprema decimalnu vrijednost temperature u bit7 i bit6 te ima preciznost od 0.25. Potrebno je očitanu vrijednost s registra 0x13 "preurediti", odbaciti podatke s ostalih bitova i pomnožiti dobivenu vrijednost s 0.25. Funkcija ne prima nikakve podatke i ne vraća nikakve podatke.

```
void rtc_get_temp()                         // funkcija za dohvacanje temperature od ds3231
{
    uint8_t MSB;
    uint8_t LSB;
    I2C_Start(Device_Write_address);            // Zapocni I2C komunikaciju sa RTC
    I2C_Write(Temperature_address);             // Zapisi koja ce se adrese citati
    I2C_Repeated_Start(Device_Read_address);    // Ponovo pokreni komunikaciju, i predaj adresu s koje se cita vrijednost
    MSB = I2C_Read_Ack();                       // Procitaj podatke
    LSB = I2C_Read_Nack();
    n1=MSB;
    n2=((LSB >> 6) * 0.25 );                    //u 0x12 je decimalni dio temperature u bit7 i bit8 (po 0.25)
    I2C_Stop();                                 // Zaustavi I2C komunikaciju
}
```

Slika 5.13. Funkcija rtc_get_temp

```
void RTC_Read_Clock(char read_clock_address)
```

Funkcija zadužena za čitanje sekunda minuta i sati sa DS3231. Očitane vrijednosti se spremaju u globalne varijable second, minute, hour. Funkcija ne vraća nikakav podatak i prima jedan podatak.

char read_clock_address – adresa s koje počinjemo čitati podatke

```
void RTC_Read_Clock(char read_clock_address)       // funkcija za dohvacanje vremena sa DS3231
{
    I2C_Start(Device_Write_address);           // Zapocni I2C komunikaciju sa RTC
    I2C_Write(read_clock_address);             // Zapisi koja ce se adrese citati
    I2C_Repeated_Start(Device_Read_address);   // Ponovo pokreni komunikaciju, i predaj adresu s koje se cita vrijednost

    second = I2C_Read_Ack();                   // Procitaj sekunde
    minute = I2C_Read_Ack();                   // Procitaj minute
    hour = I2C_Read_Nack();                    // Procitaj sat
    I2C_Stop();                                // Zaustavi I2C komunikaciju
}
```

Slika 5.14. Funkcija RTC_Read_Clock

```
void RTC_Read_Calendar(char read_calendar_address)
```

Funkcija zadužena za čitanje dana u tjednu, dana u mjesecu, mjeseca i godine sa DS3231. Te očitane vrijednosti se tim redom zapisuju u globalne varijable day, date, month, year. Funkcija ne vraća nikakav podatak i prima jedan podatak.

char read_calendar_address – adresa s koje počinjemo čitati podatke

```
void RTC_Read_Calendar(char read_calendar_address)  // funkcija za dohvacanje datuma sa DS3231
{
    I2C_Start(Device_Write_address);           // Zapocni I2C komunikaciju sa RTC
    I2C_Write(read_calendar_address);          // Zapisi koja ce se adrese citati
    I2C_Repeated_Start(Device_Read_address);   // Ponovo pokreni komunikaciju, i predaj adresu s koje se cita vrijednost

    day = I2C_Read_Ack();                      //Procitaj redni broj dana u tjednu
    date = I2C_Read_Ack();                     //Procitaj datum/dan
    month = I2C_Read_Ack();                    //Procitaj datum/mjesec
    year = I2C_Read_Nack();                    //Procitaj datum/godinu
    I2C_Stop();                                // Zaustavi I2C komunikaciju
}
```

Slika 5.15. Funkcija RTC_Read_Calendar

```
unsigned char button_state(int button)
```

Funkcija zadužena za provjeru pritiska tipki na portu B ovisno o pinu koji joj pošaljemo. Kako se kod pritiska tipki javlja “bounce” tj. registriranje više puta jednog pritiska, postavljen je delay od 30 ms sto je dovoljno za odbacivanje “lažnih pritisaka”. Ova funkcija vraća vrijednost 1 ukoliko je tipka pritisnuta ili 0 ako tipka nije pritisnuta. Funkcija prima jedan podatak.

int button – vrijednost pina, tj. tipke, koju promatramo dali je pritisnuta

```
/*  Funkcija koja provjerava dali je tipka stisnuta na portu B
    ovisno primljenoj varijabli i postavlja debeounce tipke
    Funkcija prima int vrijednost tipke koju promatramo i vraca
    1 ako je tipka stisnuta, tj 0 ako nije */

unsigned char button_state(int button)
{
    if (!(PINB & (1<<button)))
    {
        _delay_ms(30);
        if (!(PINB & (1<<button)))
            return 1;
    }
    return 0;
}
```

Slika 5.16. Funkcija button_state

```
int BCDToDecimal(int BCD)
```

Funkcija zadužena za prebacivanje iz BDC zapisa u decimalni zapis. Kako u podaci na DS3231 zapisani u BCD obliku, za lakše rukovanje koristi se ova funkcija. Funkcija vraća vrijednost u Decimalnom obliku ovisno o primljenoj vrijednosti.

```
/* Funkcija koja primljenu varijablu
    prebacuje u decimali oblik i BCD
    te tu decimalnu vrijednost vraca */

int BCDToDecimal(int BCD)                          // funkcija koja pretvara iz BCD u decimalni oblik
{
    return (((BCD>>4)*10) + (BCD & 0xF));
}
```

Slika 5.17. Funkcija BCDToDecimal

```
void screen0(void)
```

Funkcija koja ispisuje vrijeme, datum i temperaturu na početnom ekranu. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja ispisuje sve podatke na pocetnom ekranu (ekran 0)
    Funckija neprima nikakve podatke i nevraca nikakav podatak */

void screen0(void)
{
    char ds3231_buffer[20];
    snprintf(ds3231_buffer,20, "%02x:%02x", hour, minute); //zapisemo u buffer sta zelimo ispisati (x Unsigned hexadecimal integer)
    lcd_gotoxy(5,0);                                // odredujemo poziciju gdje zelimo pisati po ekranu
    lcd_puts(ds3231_buffer);                        // postavljamo na ekran informacije iz buffera

    rtc_get_temp();
    snprintf(ds3231_buffer,20,"%d.%d C", n1,n2);
    lcd_gotoxy(10,1);
    lcd_puts(ds3231_buffer);

    RTC_Read_Calendar(3);
    snprintf(ds3231_buffer,20, "%02x/%02x/%02x ", date, month, year);
    lcd_gotoxy(0,1);
    lcd_puts(ds3231_buffer);


}
```

Slika 5.18. Funkcija screen0

```
void screen01(int sati, int min)
```

Funkcija koja ispisuje vrijeme, datum i temperaturu na početnom ekranu u trenutku kada se uđe u način za prepravljanje informacija. Ova funkcija ne vraća nikakve podatke i prima sljedeće:

int sati - vrijednost sata u trenutku ulaska u način za prepravljanje informacija

int min - vrijednost minuta u trenutku ulaska u način za prepravljanje informacija.

```
/*  Funkcija koja ispisuje sve podatke na pocetnom ekranu (ekran 0)
    kada udjemo u mod za prepravljanje podataka
    Funckija prima podatke sati i min koje su tipa int
    i nevraca nikakav podatak */

void screen01(int sati, int min)
{
    char ds3231_buffer[20];
    snprintf(ds3231_buffer,20, "%02d:%02d", sati, min); //zapisemo u buffer sta zelimo ispisati (x Unsigned hexadecimal integer)
    lcd_gotoxy(5,0);                                // odredujemo poziciju gdje zelimo pisati po ekranu
    lcd_puts(ds3231_buffer);                        // postavljamo na ekran informacije iz buffera

    rtc_get_temp();
    snprintf(ds3231_buffer,20,"%d.%d C", n1,n2);
    lcd_gotoxy(10,1);
    lcd_puts(ds3231_buffer);

    RTC_Read_Calendar(3);
    snprintf(ds3231_buffer,20, "%02x/%02x/%02x ", date, month, year);
    lcd_gotoxy(0,1);
    lcd_puts(ds3231_buffer);


}
```

Slika 5.19. Funkcija screen01

```
void screen1(void)
```

Funkcija koja ispisuje sve podatke na ekranu 1. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja ispisuje sve podatke na prvom ekranu (ekran 1)
    Funckija neprima nikakve podatke i nevraca nikakav podatak*/

void screen1(void)
{
    char buffer1[20];
    lcd_gotoxy(0,0);
    lcd_puts("Rasvjeta");
    lcd_gotoxy(0,1);

    lcd_gotoxy(1,1);
    sprintf(buffer1,"%02d", hourONtime);
    lcd_puts(buffer1);

    lcd_gotoxy(3,1);
    sprintf(buffer1,":%02d", minuteONtime);
    lcd_puts(buffer1);

    lcd_gotoxy(7,1);
    lcd_puts("do");

    lcd_gotoxy(10,1);
    sprintf(buffer1,"%02d", hourOFFtime);
    lcd_puts(buffer1);

    lcd_gotoxy(12,1);
    sprintf(buffer1,":%02d", minuteOFFtime);
    lcd_puts(buffer1);


}
```

Slika 5.20. Funkcija screen1

```
void screen2(void)
```

Funkcija koja ispisuje podatke na posljednjem ekranu, ekran 2. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja ispisuje sve podatke na prvom ekranu (ekran 1)
    Funckija neprima nikakve podatke i nevraca nikakav podatak*/

void screen2(void)
{
    char buffer1[20];
    lcd_gotoxy(0,0);
    lcd_puts("Vrata:");
    lcd_gotoxy(8,0);
    sprintf(buffer1,"%02dmin", gateInterval);
    lcd_puts(buffer1);
    lcd_gotoxy(0,1);
    lcd_puts("Do:");
    lcd_gotoxy(5,1);
    sprintf(buffer1,"%02d:%02d", gateOFFtimeH,gateOFFtimeM);
    lcd_puts(buffer1);
}
```

Slika 5.21. Funkcija screen2

```
void turnON(void)
```

Funkcija zadužena za paljenje releja na portu D i C. Kako su releji upaljeni, kad na portu D ili C, pinovi poprime vrijednost 0, koristimo PORTD &=~(1<<i);. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja pali svijetla (PORT D i C)
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void turnON(void)
{
    for (uint8_t i = 2; i < 8; i++)
    {
        PORTD &=~(1<<i);
    }
    for (uint8_t j = 6; j < 8; j++)
    {
        PORTC &=~(1<<j);
    }
}
```

Slika 5.22. Funkcija turnON

```
void turnOFF(void)
```

Funkcija zadužena za gašenje releja na portu D i C. Kako su releji ugašeni, kad na portu D ili C, pinovi poprime vrijednost 1 koristimo PORTD |=(1<<i);. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja gasi svijetla (PORT D i C)
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void turnOFF(void)
{
    for (uint8_t i = 7; i >=2 ; i--)
    {
        PORTD |=(1<<i);
    }
    for (uint8_t j = 7; j >=6 ; j--)
    {
        PORTC |=(1<<j);
    }
}
```

Slika 5.23. Funkcija turnOFF

```
int intervalH (int val)
```

Funkcija zadužena za pravilno povećanje i smanjenje vrijednosti sata. Sat ne može poprimiti vrijednost veću od 24 ni manju od 01. Funkcija prima jednu varijablu int val, u tu varijablu predajemo vrijednost za koju želimo da nam se nalazi u zadanom intervalu. Ova funkcija će novo prepravljeni podatak kao takav i vratiti.

```
/*  Funkcija koja regulira da ce primljeni podatak biti
    u intervalu 0 do 24
    Funkcija prima podatak tipa int i vraca podatak tipa int */

int intervalH (int val)
{
    if ( val > 24)
    {
        val = 1;
    }
    if (val < 0)
    {
        val = 24;
    }
    return val;
}
```

Slika 5.24. Funkcija intervalH

```
int intervalM (int val)
```

Funkcija zadužena za pravilno povećanje i smanjenje vrijednosti minuta. Minute ne mogu poprimiti vrijednost veću od 59 ni manju od 00. Funkcija prima jednu varijablu int val u tu varijablu predajemo vrijednost za koju želimo da nam se nalazi u zadanom intervalu. Ova funkcija će novo prepravljeni podatak kao takav i vratiti.

```
/*  Funkcija koja regulira da ce primljeni podatak biti
    u intervalu 0 do 59
    Funkcija prima podatak tipa int i vraca podatak tipa int */

int intervalM (int val)
{
    if ( val > 59)
    {
        val = 0;
    }
    if (val < 0)
    {
        val = 59;
    }
    return val;
}
```

Slika 5.25. Funkcija intervalM

```
void gateTimer(void)
```

Funkcija koja prati dali je prošlo određeno vrijeme (vrijeme koje je zapisano u varijabli gateInterval) od trenutka kada su se vrata otvorila. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja prati dali je proslo odredjeno
     vrijeme ovisno o varijabli gateIntervalu
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void gateTimer(void)
{
    RTC_Read_Clock(0);
    int currentTime = BCDToDecimal(minute);
    int currentTimeSec = BCDToDecimal(second);
    if ( currentTime < pressedEvent)
    {
        currentTime = currentTime + pressedEvent;
    }
    if (currentTime-pressedEvent >= gateInterval)
    {
        if (currentTimeSec >= pressedEventSec)
        {
            gateFlag = false;
        }
    }
}
```

Slika 5.26. Funkcija gateTimer

```
void gateEvent(void)
```

Funkcija zadužena za zapisivanje točnog vremena kada su se vrata otvorila, tj. zatvorila. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja provjeri dali su vrata otvorena,
    ako su otvorena zapamti vrijeme i postavi zastavicu
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void gateEvent(void)
{
    if(button_state(gate))
    {
        pressedEvent = pmin;
        pressedEventSec = psec;
        gateFlag = true;
    }
}
```

Slika 5.27. Funkcija gateEvent

```
void esp_site(void)
```

Funkcija koja šalje AT komandu sa atmega16 na esp8266, kako bi uspostavili konekciju sa stranicom. AT komanda izgleda ovako AT+CIPSTART="TCP", "api.thingspeak.com",80.

Kako bi se AT komanda poslala, moraju se na kraju poslati i znakovi \r\n. Ova funkcija ne prima i ne vraća nikakve podatke.

```
/*  Funkcija koja salje s atmega16 na esp8266 preko uarta
    AT komandu za spajanje na stranicu
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void esp_site(void)
{
    char _atCommand[60];
    sprintf(_atCommand, "AT+CIPSTART=\"TCP\",\"%s\",%s", DOMAIN, PORT);
    USART_SendString(_atCommand);
    USART_SendString("\r\n");
    _delay_ms(1500);
}
```

Slika 5.28. Funkcija esp_site

```
void esp_publish(uint8_t field, uint8_t d_num, uint8_t data1, uint8_t data2)
```

Funkcija zadužena za slanje GET poruke za zapisivanje podatka na ThingSpeak stranicu. No prije slanja GET poruke, potrebno je poslati AT komandu da se odredi koliko dugačka će biti poruka. AT komanda je AT+CIPSEND=XX, gdje xx predstavlja dužinu poruke. Ova funkcija ne vraća nikakve podatke, ali prima sljedeće podatke:

uint8_t field - Polje u koje želimo upisati podatak na ThingSpeak stranici

uint8_t d_num - Broj koji kazuje koliko podataka želimo poslati

uint8_t data1 – Podatak 1 koji želimo poslati

uint8_t data2 – Podatak 2 koji želimo poslati.

```
/*  Funkcija koja salje s atmega16 na esp8266 preko uarta
    AT komandu koja ce rec kolko dugacku poruku saljemo
    i nakon toga saljemo podatak GET ..... za zapis podatka na stranici
    Funkcija prima podatake i svi su tipa uint8_t
    field - koje polje na stranici thingspeak se upisuje podatak
    d_num - kolko podataka ce se zapisati u polje
    data1 - podatak 1 koji se upisuje u polje
    data2 - podatak 2 koji se upisuje u polje
    Funkcija nevraca nikakav podatak */

void esp_publish(uint8_t field, uint8_t d_num, uint8_t data1, uint8_t data2)
{
    char _atCommand[60];
    char get_buffer[150];
    if (d_num == 1)
    {
        sprintf(get_buffer, "GET https://api.thingspeak.com/update?api_key=%s&field%d=%d",API_WRITE_KEY,field,data1);
    }
    else if (d_num ==2)
    {
        sprintf(get_buffer, "GET https://api.thingspeak.com/update?api_key=%s&field%d=%d%d",API_WRITE_KEY,field,data1,data2);
    }

    sprintf(_atCommand, "AT+CIPSEND=%d", (strlen(get_buffer)+2));
    USART_SendString(_atCommand);
    USART_SendString("\r\n");
    _delay_ms(1500);
    USART_SendString(get_buffer);
    USART_SendString("\r\n");
    _delay_ms(1500);
}
```

Slika 5.29. Funkcija esp_publish

```
void esp_local_update(uint8_t field)
```

Funkcija zadužena za slanje GET poruke za čitanje podatka s ThingSpeak stranice. No, prije slanja GET poruke, potrebno je poslati AT komandu da se odredi koliko dugačka će biti poruka. AT komanda je AT+CIPSEND=XX, gdje xx predstavlja dužinu poruke. Ova funkcija ne vraća nikakve podatke, ali prima sljedeći podatak

uint8_t field – Koje polje želimo pročitati s ThingSpeak stranice.

```
/*  Funkcija koja salje s atmega16 na esp8266 preko uarta
    AT komandu koja ce rec kolko dugacku poruku saljemo
    i nakon toga saljemo podatak GET ..... za citanje podatka sa stranice
    Funkcija prima podatak tipa uint8_t
    field - koje polje na stranici thingspeak se upisuje podatak
    Funkcija nevraca nikakav podatak */

void esp_local_update(uint8_t field)
{
    char _atCommand[60];
    char get_buffer[150];
    sprintf(get_buffer, "GET https://api.thingspeak.com/channels/%s/fields/%d/last.txt",CHANNEL_ID,field);
    sprintf(_atCommand, "AT+CIPSEND=%d", (strlen(get_buffer)+2));

    USART_SendString(_atCommand);
    USART_SendString("\r\n");
    _delay_ms(1500);
    USART_SendString(get_buffer);
    USART_SendString("\r\n");

}
```

Slika 5.30. Funkcija esp_local_update

```
void uart_primi(uint8_t field,char* ocekivani_odgovor,char* neocekivani_odgovor)
```

Funkcija koja prima poruku sa esp8266 preko uart-a, sve dok se u primljenoj poruci ne pronađe određena poruka. Primljena poruka se sprema u uart_poruka. Ako se u poruci nalazi ERROR, onda će se obrisati svi primljeni podaci i pointer ret će se postaviti u 0. Ova funkcija ne vraća nikakav podatak ali prima sljedeće

uint8_t field – mjesto gdje želimo zapisati podatak u funkciji uart_korisni_dio

char* ocekivani_odgovor – Odgovor koji očekujemo ako je uspješno poslan podatak na esp8266

char* neocekivani_odgovor - Odgovor koji očekujemo ako nije uspješno poslan podatak na esp8266

```
/*  Funkcija koja prima poruku s esp8266 preko uarta,
    (primljena poruka se nalazi u uart_poruka),
    Funkcija prima podatak tipa uint8_t i char*
    field - koje polje na stranici thingspeak se upisuje podatak
    ocekivani_odgovor - podatak koji ocekujemo ukoliko smo uspjesno poslali poruku
    neocekivani_odgovor + podatak koji ocekujemo ukoliko smo neuspjesno poslali poruku
    Funkcija nevraca nikakav podatak */

void uart_primi(uint8_t field,char* ocekivani_odgovor,char* neocekivani_odgovor)
{
    uint8_t i=0;
    //primaj poruku sve dok nedodjes do ocekivani_odgovor ili neocekivani_odgovor
    do
    {
        uart_poruka[i]=USART_Recieve();
        i++;
        if (i==150)
        {
            i=0;
        }
    }while ((strstr(uart_poruka,ocekivani_odgovor)==NULL)&&(strstr(uart_poruka,neocekivani_odgovor)==NULL));


    if (strstr(uart_poruka,"ERROR")!=NULL)
    {
        memset(uart_poruka,0,150);
        ret =0;
    }
    uart_korisni_dio(field);
}
```

Slika 5.31. Funkcija uart_primi

```
void uart_korisni_dio(uint8_t field)
```

Funkcija koja provjerava šta se nalazi u varijabli uart_poruka i ovisno o tome odlučuje šta će napraviti. Nakon toga, iz cijele poruke uzima samo koristan dio uart_poruke i zapisuje je u lokalne varijable, za podešavanje sustava ili postavlja zastavicu za wifi koja pokazuje da li esp8266 ima wifi konekciju. Ova funkcija ne vraća nikakav podatak ali prima jedan.

uint8_t field – ovisno o ovom broju određuje se gdje treba zapisati podatak

```
/*  Funkcija koja ovisno o primljenoj poruci s esp8266 preko uarta,
    (primljena poruka se nalazi u uart_poruka),
    obnavlja vrijednosti lokalnih varijabli ili postavlja flag za wifi,
    Funkcija prima podatak tipa uint8_t
    field - koje polje na stranici thingspeak se upisuje podatak
    Funkcija nevraca nikakav podatak */

void uart_korisni_dio(uint8_t field)
{
    if (strstr(uart_poruka,"CLO")!=NULL)
    {
        ret = strstr(uart_poruka,"+IPD");   // uzmi adresu gdje pocinje "+IPD" u pristigloj poruci
        int vrijednost=atoi(ret+7);         //korisni podatak se nalazi na +7 adresi od ret +IPD

        //provjeravamo gdje moramo upisati podatak
        if (field==1)
        {
            hourONtime=vrijednost/100;
            minuteONtime=vrijednost%100;
        }
        else if (field==2)
        {
            hourOFFtime=vrijednost/100;
            minuteOFFtime=vrijednost%100;
        }
        else if (field==3)
        {
            gateInterval=vrijednost;
        }
        else if (field==4)
        {
            gateOFFtimeH=vrijednost/100;
            gateOFFtimeM=vrijednost%100;
        }
        //ocistimo uart_poruka i ret
        memset(uart_poruka,0,150);
        ret =0;
    }
    else if (strstr(uart_poruka,"STATUS:")!=NULL)
    {
        ret = strstr(uart_poruka,"STATUS:");
        int vrijednost = atoi(ret+7);

        if (vrijednost==5)
            wifi_stanje=false;
        else
            wifi_stanje=true;
        //ocistimo uart_poruka i ret
        memset(uart_poruka,0,150);
        ret =0;
    }
    else if (strstr(uart_poruka,"WIFI CONNECTED")!=NULL)
    {
        wifi_stanje=true;
        //ocistimo uart_poruka i ret
        memset(uart_poruka,0,150);
        ret =0;
    }
    else if (strstr(uart_poruka,"FAIL")!=NULL)
    {
        wifi_stanje=false;
        //ocistimo uart_poruka i ret
        memset(uart_poruka,0,150);
        ret =0;
    }
}
```

Slika 5.32. Funkcija uart_korisni_dio

```
void esp_wifi(void)
```

Funkcija koja šalje AT komandu sa ATmega16 na esp8266 za spajanje na wifi mrežu. AT komanda glasi AT+CWJAP=”SSID”,”PASSWORD”. Funkcija ne vraća nikakav podatak i ne prima nikakav podatak.

```
/* Funkcija koja salje s atmega16 na esp8266 preko uarta
    AT komandu za spajanje na wifi mrezu
    Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void esp_wifi(void)
{
    char _atCommand[60];
    sprintf(_atCommand, "AT+CWJAP=\"%s\",\"%s\"", SSID, PASSWORD);
    USART_SendString(_atCommand);
    USART_SendString("\r\n");
    uart_primi(0,"WIFI CONNECTED","FAIL");
    _delay_ms(2000);
}
```

Slika 5.33. Funkcija esp_wifi

5.4 Izrada mobilne aplikacije

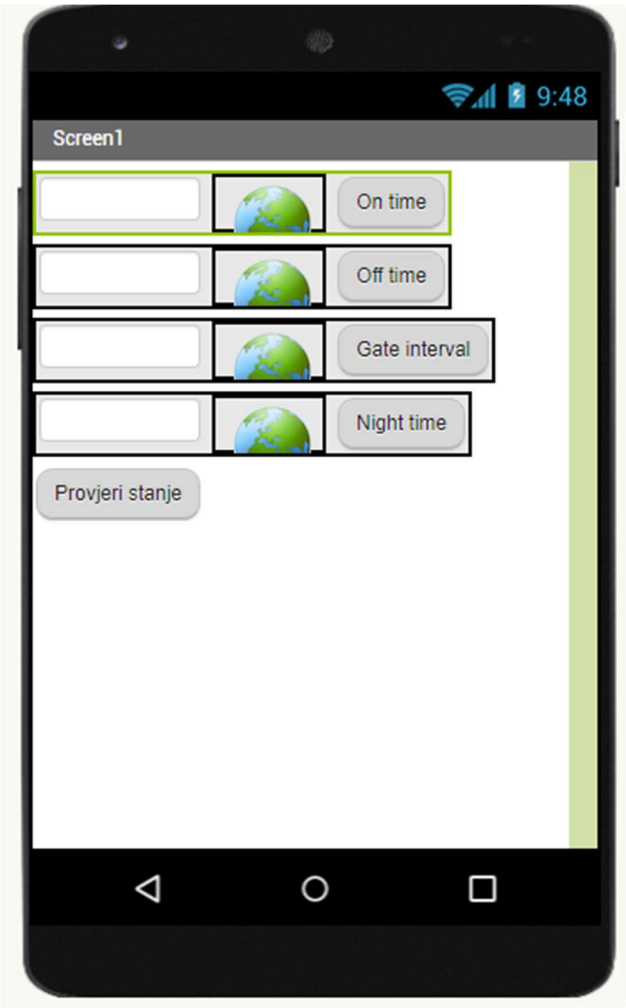
Mobilna aplikacija za upravljanje sustavom rasvjete napravljena je korištenjem Mit App Inventor platforme.

Kreiranje aplikacije započinje odabirom naziva aplikacija. Nakon toga otvara se vizualno sučelje preko kojeg se izabiru elementi koji će se koristiti u aplikaciji.

Za ovaj diplomski rad naziv mobilne aplikacije je Rasvjeta, te su korišteni sljedeći elementi: Horizontal Arrangement, Text Box, Web Viewer, Button i Web. Text Box u ovoj aplikaciji služi za unos vrijednosti koju želimo poslati na ThingSpeak. Button služi za slanje unešenog podatka, i za primanje podatka sa ThingSpeak platforme. Web Viewer služi za prikaz primljenog podatka sa ThingSpeak platforme. Horizontal Arragement služi kako bi se omogućilo horizontalno slaganje elemenata u aplikaciji. Web element je takozvani „nevidljivi“ element, koji omogućuje spajanje mobitela na Internet.

Pošto sustav ima 4 vrijednosti koje je moguće promijeniti, potrebno je bilo napraviti po 4 od skoro svakog gore navedenog elementa. Iznimka je Button, kojih treba biti 5 i jedan Web. Nakon postavljanja svih potrebnih elemenata i uređivanja njihovih parametra, aplikacija će izgledati kao na slici 5.34.

Poslije postavljanja elementa potrebno je prijeći u Blocks način rada, gdje će se povezati svi elementi i isprogramirati logika aplikacije. Aplikacija je isprogramirana tako, da pritiskom na tipke „On time“, „Off time“, „Gate interval“ i „Night time“, vrijednost iz Text Box od odgovarajuće tipke će se poslati na odgovarajuće polje na ThingSpeak platformi. Ukoliko se stisne tipka „Provjeri stanje“, očitati će se sve vrijednosti s ThingSpeak platforme i zapisati unutar Web Viewer-a, pored odgovarajućeg polja unutar aplikacije. Ovaj „kod“ u blokovskom načinu je prikazan na slici 5.35.



Slika 5.34. Izgled aplikacije unutar Mit App Inventor platforme

Slika 5.35. Blokovski kod mobilne aplikacije

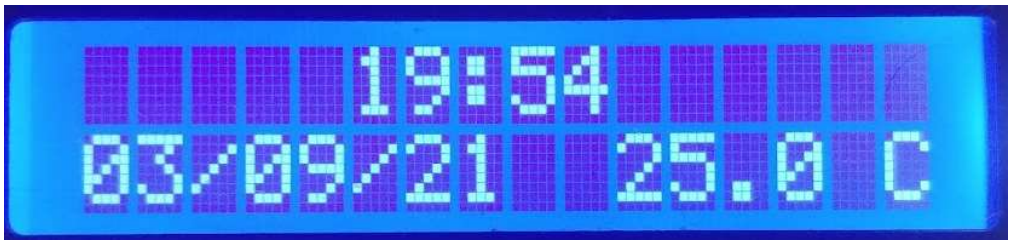
6. TESTIRANJE SUSTAVA

6.1 Korištenje sustava

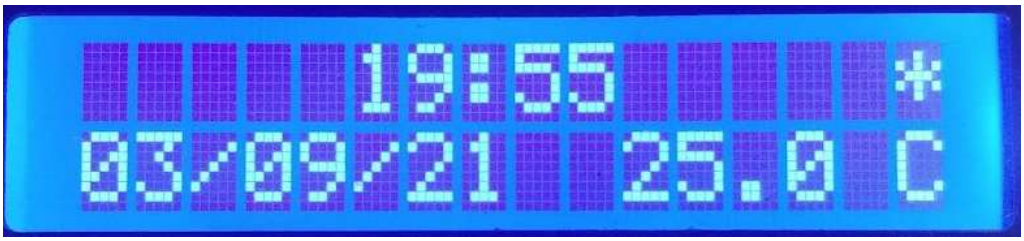
Ručno kontroliranje sustava se vrši preko lcd-a i tipki. Sustav ima 6 tipki. Tipke: naprijed, nazad, plus, minus, uredi i postavi. Za kretanje između različitih ekrana i odabira drugih vrijednosti, koje želimo promijeniti kada je sustav u načinu rada za izmjenu podataka, koristimo tipke naprijed i nazad. Tipke plus i minus, isključivo se koriste kada je sustav u načinu rada za ispravljanje podataka. Tipka uredi, služi za ulaz u način rada za ispravljanje podataka, dok tipka postavi, služi za izlazak iz tog načina rada. Sustav se također može kontrolirati i preko mobilne aplikacije, što je objašnjeno u sljedećem poglavlju.

Prilikom paljenja sustava, sustav će se odmah pokušati spojiti na wifi mrežu, nakon toga na zaslonu će se pokazati stanje ekrana 0, tj. stanje u kojem sustav prikazuje sate, minute, datum, i temperaturu. Kada se na ovom ekranu pojavi simbol *, sustav je ušao u način rada za primanje podataka kada će sustav s ThingSpeak platforme preuzeti sve podatke. Na ekranu 1 se prikazuju informacije za sustav, kada se on nalazi u intervalu za paljenje i gašenje rasvjete. Na ekranu 2 su prikazane informacije, koliko dugo će svjetlo biti upaljeno, kada se vrata otvore i do kada treba trajati interval, u kojem će sustav reagirati na otvaranje vrata. Ukoliko se na bilo kojem od ovih ekrana pritisne tipka „uredi“, na ekranu će se pojaviti simbol < ili > koji će pokazivati koju vrijednost trenutno možemo promijeniti. Pritiskom na tipku postavi, na ekranu će se ispisati „UPDATING“ kada će sustav poslati sve podatke s trenutnog ekrana na platformu ThingSpeak.

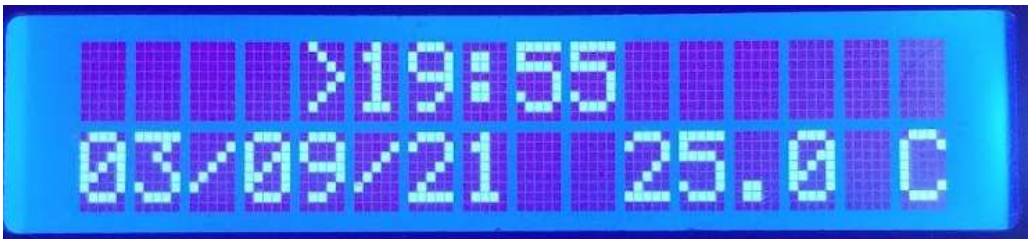
Na slikama 6.1, 6.2 i 6.3 prikazan je početni zaslon u svim načinima rada. Na slikama 6.4, 6.5 i 6.6 su prikazana stanja ekrana 1, 2 i stanje kada sustav šalje podatke na ThingSpeak platformu.



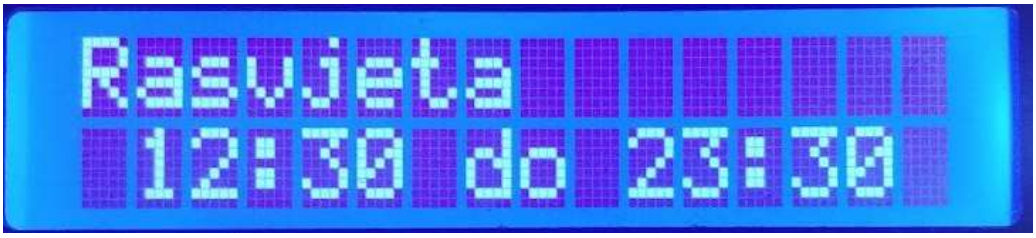
Slika 6.1. Prikaz nultog stanja ekrana



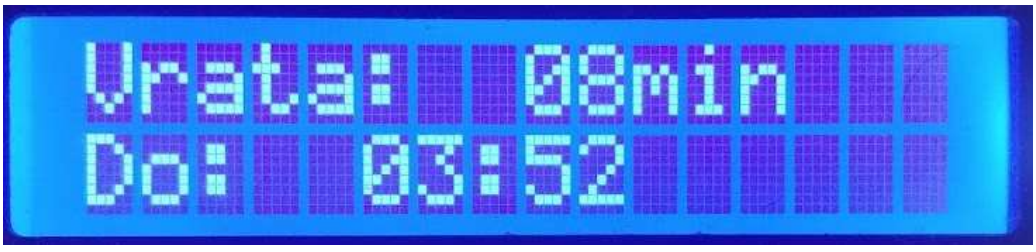
Slika 6.2. Prikaz nultog stanja ekrana kada prima podatke sa platforme ThingSpeak



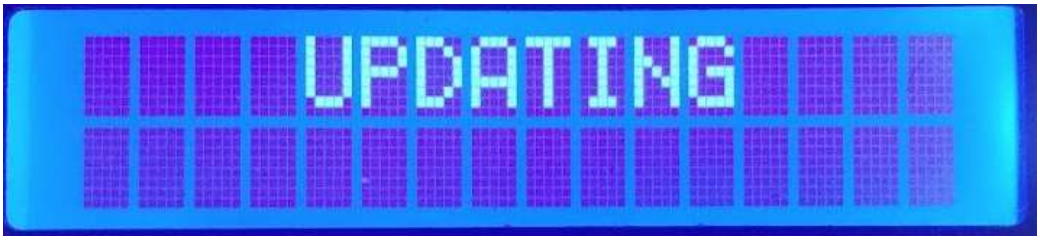
Slika 6.3. Prikaz nultog stanja ekrana u načinu rada za izmjenu podataka



Slika 6.4. Prikaz prvog stanja ekrana



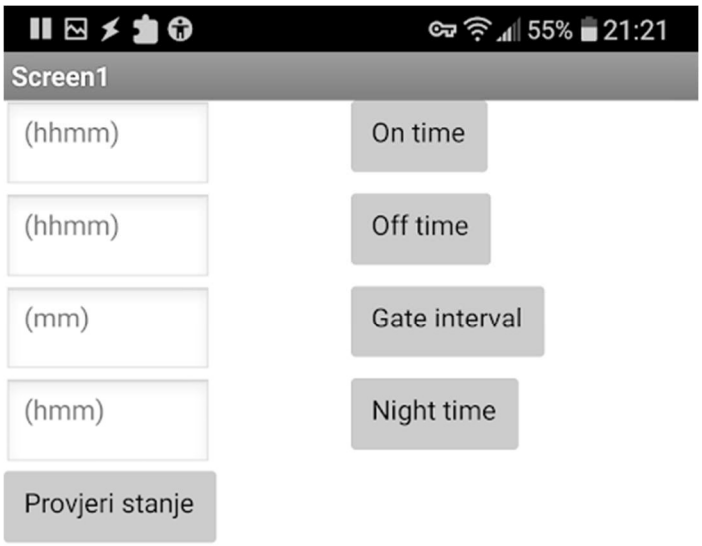
Slika 6.5. Prikaz drugog stanja ekrana



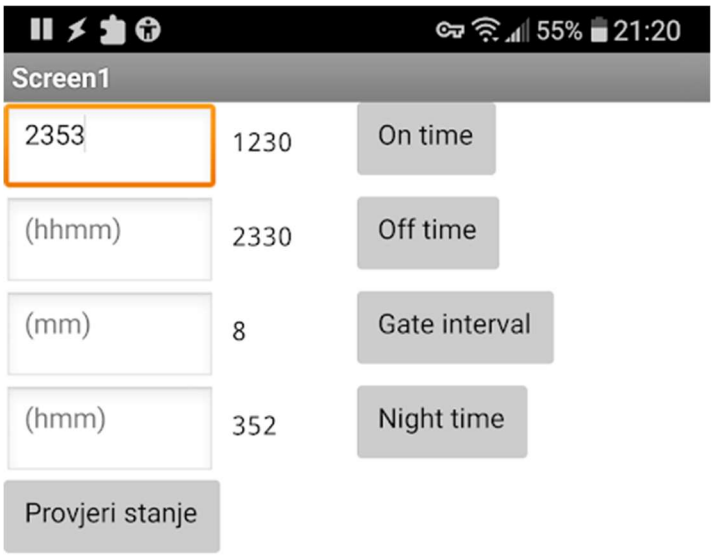
Slika 6.6. Prikaz ekrana kada šalje podatke na ThingSpeak platformu

6.2 Korištenje mobilne aplikacije

Kako bi se upravljalo sustavom rasvjete putem interneta potrebno je koristiti aplikaciju za mobitel, naziva „Rasvjeta“. Pokretanjem aplikacije prikazati će se na ekranu 4 polja i 5 tipki. Na tipkama piše „On time“, „Off time“ , „Gate interval“, „Night time“ i „Provjeri stanje“. Pored tipki se nalaze polja u kojima piše kojeg formata mora biti podatak koji želimo poslati, pa tako za On time piše (hhmm) dok za Gate interval piše (mm). To nam govori kako On time mora primiti sate i minute bez razmaka, primjer 1253 bi bilo 12 sati i 53 minute. Kada se u željeno polje unese podatak i pritisne odgovarajuća tipka za to polje, uneseni podatak će se poslati na ThingSpeak platformu, koji će sustav kasnije prihvatiti s iste platforme. Ukoliko se pritisne tipka „Provjeri stanje“, pored svakog polja će se prikazati koje je trenutno stanje podataka na platformi ThingSpeak odnosno na sustavu za rasvjetu. Na slici 6.7. je prikazano kako izgleda aplikacija kada se prvi puta uđe u nju, a slika 6.8. prikazuje kako izgleda poslije pritiska tipke „Provjeri stanje“ i upisa jednog podatka.



Slika 6.7. Mobilna aplikacija pri provom pokretanju



Slika 6.8. Mobilna aplikacija poslije pritiska na tipku „Provjeri stanje“ i upisa podatka u polje za On time

6.3 Uočeni problemi sustava

Pri radu sustava uočeni su problemi. Zbog ograničenja brzine platforme ThingSpeak za slanje i preuzimanje podataka, sustav će se “zaustavit“ dok radi slanje i primanje podataka. Kako sustav svake minute radi primanje podataka može se desiti da trenutak otvaranja vrata se preklopi sa trenutkom kada se vrši primanje podataka, što će rezultirati s time da sustav neće reagirati na otvaranje vrata. Ovaj problem se može umanjiti tako da se napravi rjeđe preuzimanje podataka s platforme ThingSpeak dok se sustav nalazi u intervalu kada očekuje otvaranje vrata.

ZAKLJUČAK

U ovom diplomskom radu napravljen je sustav za upravljanje rasvjetom putem WiFi-a, koji je moguće kao takav koristiti u kućanstvu ili nekim drugim privatnim ili javnim prostorima.

Ovim sustavom korisnik može upravljati sa mjesta lokacije sustava, korištenjem ugrađenog zaslona i tipkovnice. Ovaj način upravljanja je posebno koristan kada internet nije dostupan. No, ukoliko je sustav spojen na internet preko WiFi mreže, upravljati se može preko mobitela korištenjem mobilne aplikacije. Mobilna aplikacija omogućuje brze izmjene postavki sustava i kada korisnik nije blizu samog sustava.

Pogodnosti ovog sustava su njegova niska cijena te pouzdanost koju pruža čip DS3231 sa svojom integriranom baterijom. Također korištenjem platforme ThingSpeak sve promjene sustava su sigurno spremljene u “oblaku”. Pri nestanku napajanja DS3231 će nastaviti vjerno pratiti vrijeme, a ThingSpeak platforma će osigurati da sustav pri povratku napajanja može preuzeti zadnje postavke sustava.

LITERATURA

[1] https://www.philips-hue.com/en-us, 8.9.2021

[2] https://wyze.com/, 8.9.2021

[3] http://ww1.microchip.com/downloads/en/devicedoc/doc2466.pdf, 1.9.2021.

[4] https://www.theengineeringprojects.com/2018/06/introduction-to-atmega16.html, 1.9.2021.

[5] https://www.espressif.com/en/products/socs/esp8266, 1.9.2021.

[6] https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/#AT, 1.9.2021.

[7] https://en.wikipedia.org/wiki/ESP8266, 1.9.2021.

[8] http://wiki.sunfounder.cc/index.php?title=8_Channel_5V_Relay_Module, 1.9.2021.

[9] https://datasheets.maximintegrated.com/en/ds/DS3231.pdf, 1.9.2021. [8] [10]https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/, 1.9.2021.

[11] https://www.electronicsforu.com/resources/learn-electronics/16x2-lcd-pinout-diagram, 1.9.2021.

[12] http://www.hlktech.net/product_detail.php?ProId=60, 1.9.2021.

[13] https://www.fischl.de/usbasp/, 1.9.2021.

[14] https://d3s11pzv7w3h1q.cloudfront.net/wp-content/uploads/AC-PG-USBASP.jpg, 1.9.2021.

[15] https://microchipdeveloper.com/atstudio:studio7intro, 1.9.2021.

[16] https://thingspeak.com/, 1.9.2021.

[17] https://appinventor.mit.edu/, 1.9.2021.

[18] https://www.electronicwings.com/avr-atmega/atmega1632-i2c, 1.9.2021.

[19] Vježba 5. Kristalni oscilator i Biblioteke: LCD i WDT, Primjena Mikroupravljačkih Sustava – laboratorijske vježbe, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Zavod za komunikacije.

SAŽETAK

Naslov: Upravljanje rasvjetom putem WiFi-a

U ovome diplomskom radu opisana je izrada sustava za upravljanje rasvjetom putem WiFi-a, ručno upravljanje i izrada mobilne aplikacije. Prikazano je kako korištenjem jeftinih komponenti i besplatnim programskim alatima je moguće napraviti ovakav sustav. Iako se u ovom diplomskom radu koriste 8 releja za kontroliranje rasvjete, uz minimalne modifikacije sustava, sustav se može proširiti kako bi se omogućilo kontroliranje vise releja. Također kontrola ovog sustava trenutno se može kontrolirati preko mobilne aplikacije za Android sustav no mogu se napraviti aplikacije i za druge operativne sustave bez mijenjanja trenutnog sustava za upravljanje rasvjetom.

Ključne riječi: Upravljanje rasvjetom putem WiFi-a, esp8266, ATmega16, mikrokontroler, ds3231 mobilna aplikacija, ThingSpeak.

ABSTRACT

Title: Lighting control via WiFi

This thesis describes the development of lighting control systems via WiFi, manual control and the development of a mobile application. It is shown how it is possible to make such a system by using cost efficient components and free software tools. Although 8 lighting control relays are used in this thesis, with minimal system modifications, the system can be expanded to allow multiple relays to be controlled. Also the control of this system can currently be controlled via a mobile application for the Android system but applications can also be made for other operating systems without changing the current lighting management system.

Keywords: Lighting control via WiFi, esp8266, ATmega16, microcontroller, ds3231 mobile application, ThingSpeak.

ŽIVOTOPIS

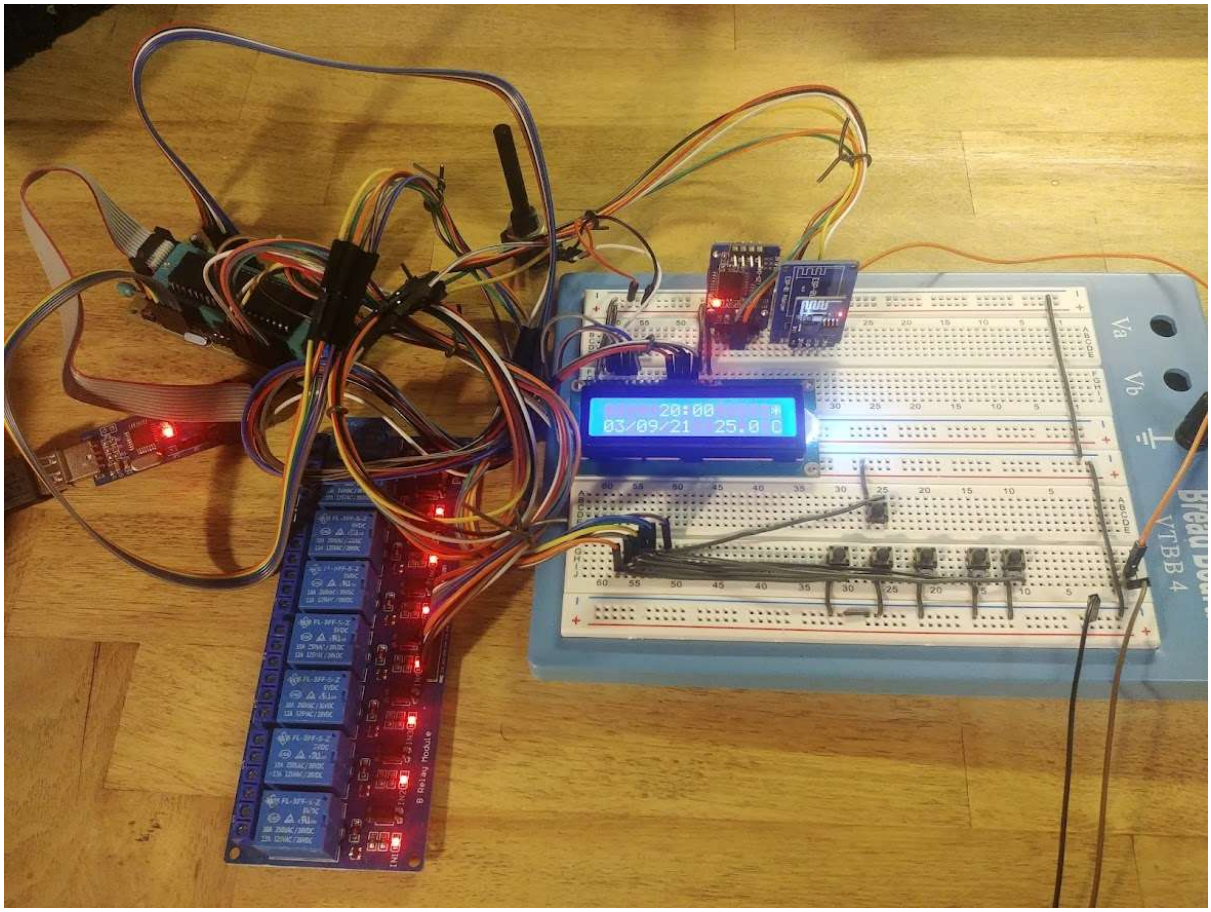
Vedran Krušarovski rođen je 4.1.1993. godine u Osijeku. Od rođenja do svoje 5-te godine živio je u Osijeku nakon čega se preselio u Beli Manastir. Osnovnoškolsko obrazovanje je stekao u Belom Manastiru u školi Dr. Franje Tuđmana. Nakon Osnovne škole upisuje Prvu srednju školu Beli Manastir, smjer elektrotehničar, gdje je sve godine prošao s odličnim uspjehom i dobio nagradu učenika škole. 2012. Godine upisuje preddiplomski sveučilišni studij Elektrotehnike na Elektrotehničkom fakultetu u Osijeku.

Preddiplomski studij elektrotehnike završava 2018. godine sa završnim radom „Internet prekidač“ i iste godine upisuje diplomski studij elektrotehnike, smjera „Komunikacije i informatika“ i odjela „DKA – Komunikacijske tehnologije“. Stručnu praksu odrađuje u tvrtki „Electronic Center d.o.o.“ na poziciji „Asistent u prodaji elektroničkih i informatičkih komponenti“.

PRILOZI

Slike

Slika 1. prikazuje gotov sustav za kontroliranje rasvjete putem WiFi-a



Slika 1 sustav za kontroliranje rasvjete putem WiFi-a

PROGRAMSKI KOD

main.c

```
/*
 * lcdDS3231.c
 *
 * Created: 22/02/2021 09:34:00
 * Author : vkrus
 */
#define  F_CPU 11059200UL
#defineprev   0      //definiranje tipki koje cu koristiti u kodu
#define next  1
#define edit  2
#define set   3
#define plus  4
#define minus 5
#define gate  6

#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "I2C_Master_H_file.h"
#include "lcd.h"
#include <util/delay.h>
#include <avr/interrupt.h>
#include "Uart.h"
#include <stdlib.h>

#define SSID                    "SSID"
#define PASSWORD                "PASSWORD"
#define DOMAIN                  "api.thingspeak.com"
#define PORT                    "80"
#define API_WRITE_KEY           "BLPU***********"
#define CHANNEL_ID              "14*****"

#define Device_Write_address    0xD0   // Adresa od DS3231 za zapis podataka
#define Device_Read_address     0xD1   // Adresa od DS3231 za citanje podataka
#define Temperature_address     0x11   // Adresa od DS3231  na kojoj se nalazi
temperatura
#define hour_24                                 0x00

int second,minute,hour,day,date,month,year,n1,n2;      // Inicijalizacija varijabli
uint8_t hourONtime = 23;    // vrijeme(sat) kada se sustav pali
uint8_t minuteONtime = 25;  // vrijeme(minute) kada se sustav pali
uint8_t hourOFFtime = 23;   // vrijeme(sat) kada se sustav gasi
uint8_t minuteOFFtime = 30; // vrijeme(minute) kada se sustav gasi
uint8_t gateOFFtimeH = 3;   // vrijeme(sat) do kada ce sustav provjeravat dali su
vrata otvorena
uint8_t gateOFFtimeM = 30;  // vrijeme(minute) do kada ce sustav provjeravat dali su
vrata otvorena
uint8_t screenPicker = 0;   // varijabla koja omogucuje listanje na ekranu
uint8_t editMenu = 0;       // varijabla koja omogucuje odabiranje sta zelimo
prepraviti
bool light = false;         // prati dali je svijetlo upaljeno ili ugaseno
bool gateFlag = false;            // prati dali su vrata otvore (bila otvorene)
uint8_t pressedEvent = 0;   // varijabla za spremanje trenutka(minute) kada su vrata
bila otvorena
uint8_t pressedEventSec = 0;       // varijabla za spremanje trenutka(sekunde) kada su
vrata bila otvorena
```

```
uint8_t gateInterval= 3;    // kolko dugo u minutama zelimo da svijetlo bude upaljeno
ako su se vraata otvorila
uint8_t psat = 0;           // ovdje zapisemo sat kako bi kod bio lakse za citat
uint8_t pmin = 0;                  // ovdje zapisemo minute kako bi kod bio lakse za
citat
uint8_t psec = 0;
char uart_poruka[150];
char odgovor[10];
char *ret;
bool wifi_stanje;

void RTC_Clock_Write(char _hour, char _minute, char _second, char AMPM)
{
       _hour |= AMPM;
       I2C_Start(Device_Write_address);   // Zapocni I2C komunikaciju sa RTC
       I2C_Write(0);               // Zapisi na 0 lokaciju, za vrijednosti sekunde
       I2C_Write(_second);         // Zapisi na lokaciju 00 koja prima vrijednost
sekunde
       I2C_Write(_minute);         // Zapisi na lokaciju 01 koja prima vrijednost
minute
       I2C_Write(_hour);           // Zapisi na lokaciju 02 koja prima vrijednost sate
       I2C_Stop();                              // Zaustavi I2C komunikaciju
}

void RTC_Calendar_Write(char _day, char _date, char _month, char _year)
{
       I2C_Start(Device_Write_address);   // Zapocni I2C komunikaciju sa RTC
       I2C_Write(3);         // Zapisi na 3 lokaciju, za vrijednosti dana
       I2C_Write(_day);      // Zapisi na lokaciju 03 koja prima vrijednost dana, redni
broj dana u tjednu
       I2C_Write(_date);     // Zapisi na lokaciju 04 koja prima vrijednost datuma/dan
       I2C_Write(_month);    // Zapisi na lokaciju 05 koja prima vrijednost
datuma/mjesec
       I2C_Write(_year);     // Zapisi na lokaciju 06 koja prima vrijednost
datum/godina
       I2C_Stop();                 // Zaustavi I2C komunikaciju
}

void rtc_get_temp()  // funkcija za dohvacanje temperature od ds3231
{
       uint8_t MSB;
       uint8_t LSB;
       I2C_Start(Device_Write_address);   // Zapisi koja ce se adrese citati
       I2C_Repeated_Start(Device_Read_address); // Ponovo pokreni komunikaciju, i
predaj adresu s koje se cita vrijednost
       MSB = I2C_Read_Ack();                    // Procitaj podatke
       LSB = I2C_Read_Nack();
       n1=MSB;
       n2=((LSB >> 6) * 0.25 );    //u 0x12 je decimalni dio temperature u bit7 i bit8
(po 0.25)
       I2C_Stop();                              // Zaustavi I2C komunikaciju
}

void RTC_Read_Clock(char read_clock_address)     // funkcija za dohvacanje vremena sa
DS3231
{
       I2C_Start(Device_Write_address);                 // Zapocni I2C komunikaciju sa
RTC
       I2C_Write(read_clock_address);                          // Zapisi koja ce se
adrese citati
       I2C_Repeated_Start(Device_Read_address); // Ponovo pokreni komunikaciju, i
predaj adresu s koje se cita vrijednost
```

```
	second = I2C_Read_Ack();                                    // Procitaj sekunde
	minute = I2C_Read_Ack();                                    // Procitaj minute
	hour = I2C_Read_Nack();                                             // Procitaj sat
	I2C_Stop();                                                                 //
Zaustavi I2C komunikaciju
}

void RTC_Read_Calendar(char read_calendar_address)  // funkcija za dohvacanje datuma
sa DS3231
{
	I2C_Start(Device_Write_address);                // Zapocni I2C komunikaciju sa
RTC
	I2C_Write(read_calendar_address);               // Zapisi koja ce se adrese
citati
	I2C_Repeated_Start(Device_Read_address); // Ponovo pokreni komunikaciju, i
predaj adresu s koje se cita vrijednost

	day = I2C_Read_Ack();                                               //Procitaj redni
broj dana u tjednu
	date = I2C_Read_Ack();                                              //Procitaj
datum/dan
	month = I2C_Read_Ack();                                             //Procitaj
datum/mjesec
	year = I2C_Read_Nack();                                             //Procitaj
datum/godinu
	I2C_Stop();                                                                 //
Zaustavi I2C komunikaciju
}

/*	Funkcija koja provjerava dali je tipka stisnuta na portu B
	ovisno primljenoj varijabli i postavlja debeounce tipke
	Funkcija prima int vrijednost tipke koju promatramo i vraca
	1 ako je tipka stisnuta, tj 0 ako nije */

unsigned char button_state(int button)
{
	if (!(PINB & (1<<button)))
	{
		_delay_ms(30);
		if (!(PINB & (1<<button)))
			return 1;
	}
	return 0;
}

/* Funkcija koja primljenu varijablu
	prebacuje u decimali oblik i BCD
	te tu decimalnu vrijednost vraca */

int BCDToDecimal(int BCD)                                           // funkcija koja
pretvara iz BCD u decimalni oblik
{
	return (((BCD>>4)*10) + (BCD & 0xF));
}

/*	Funkcija koja ispisuje sve podatke na pocetnom ekranu (ekran 0)
	Funckija neprima nikakve podatke i nevraca nikakav podatak */

void screen0(void)
{
	char ds3231_buffer[20];
```

```
	snprintf(ds3231_buffer,20, "%02x:%02x", hour, minute); //zapisemo u buffer sta
zelimo ispisati (x Unsigned hexadecimal integer)
	lcd_gotoxy(5,0);                                                    //
odredujemo poziciju gdje zelimo pisati po ekranu
	lcd_puts(ds3231_buffer);                                          //
postavljamo na ekran informacije iz buffera

	rtc_get_temp();
	snprintf(ds3231_buffer,20,"%d.%d C", n1,n2);
	lcd_gotoxy(10,1);
	lcd_puts(ds3231_buffer);

	RTC_Read_Calendar(3);
	snprintf(ds3231_buffer,20, "%02x/%02x/%02x ", date, month, year);
	lcd_gotoxy(0,1);
	lcd_puts(ds3231_buffer);


}

/*	Funkcija koja ispisuje sve podatke na pocetnom ekranu (ekran 0)
	kada udjemo u mod za prepravljanje podataka
	Funckija prima podatke sati i min koje su tipa int
	i nevraca nikakav podatak */

void screen01(int sati, int min)

{
	char ds3231_buffer[20];
	snprintf(ds3231_buffer,20, "%02d:%02d", sati, min); //zapisemo u buffer sta
zelimo ispisati (x Unsigned hexadecimal integer)
	lcd_gotoxy(5,0);                                                    //
odredujemo poziciju gdje zelimo pisati po ekranu
	lcd_puts(ds3231_buffer);                                  // postavljamo na
ekran informacije iz buffera

	rtc_get_temp();
	snprintf(ds3231_buffer,20,"%d.%d C", n1,n2);
	lcd_gotoxy(10,1);
	lcd_puts(ds3231_buffer);

	RTC_Read_Calendar(3);
	snprintf(ds3231_buffer,20, "%02x/%02x/%02x ", date, month, year);
	lcd_gotoxy(0,1);
	lcd_puts(ds3231_buffer);


}

/*	Funkcija koja ispisuje sve podatke na prvom ekranu (ekran 1)
	Funckija neprima nikakve podatke i nevraca nikakav podatak*/

void screen1(void)
{
	char buffer1[20];
	lcd_gotoxy(0,0);
	lcd_puts("Rasvjeta");
	lcd_gotoxy(0,1);

	lcd_gotoxy(1,1);
	sprintf(buffer1,"%02d", hourONtime);
	lcd_puts(buffer1);
```

```
	lcd_gotoxy(3,1);
	sprintf(buffer1,":%02d", minuteONtime);
	lcd_puts(buffer1);

	lcd_gotoxy(7,1);
	lcd_puts("do");

	lcd_gotoxy(10,1);
	sprintf(buffer1,"%02d", hourOFFtime);
	lcd_puts(buffer1);

	lcd_gotoxy(12,1);
	sprintf(buffer1,":%02d", minuteOFFtime);
	lcd_puts(buffer1);


}

/*	Funkcija koja ispisuje sve podatke na prvom ekranu (ekran 1)
	Funckija neprima nikakve podatke i nevraca nikakav podatak*/

void screen2(void)
{
	char buffer1[20];
	lcd_gotoxy(0,0);
	lcd_puts("Vrata:");
	lcd_gotoxy(8,0);
	sprintf(buffer1,"%02dmin", gateInterval);
	lcd_puts(buffer1);
	lcd_gotoxy(0,1);
	lcd_puts("Do:");
	lcd_gotoxy(5,1);
	sprintf(buffer1,"%02d:%02d", gateOFFtimeH,gateOFFtimeM);
	lcd_puts(buffer1);
}

/*	Funkcija koja pali svijetla (PORT D i C)
	Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void turnON(void)
{
	for (uint8_t i = 2; i < 8; i++)
	{
		PORTD &=~(1<<i);
	}
	for (uint8_t j = 6; j < 8; j++)
	{
		PORTC &=~(1<<j);
	}
}

/*	Funkcija koja gasi svijetla (PORT D i C)
	Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void turnOFF(void)
{
	for (uint8_t i = 7; i >=2 ; i--)
	{
		PORTD |=(1<<i);
	}
	for (uint8_t j = 7; j >=6 ; j--)
```

```
        {
                PORTC |=(1<<j);
        }
}

/*      Funkcija koja regulira da ce primljeni podatak biti
        u intervalu 0 do 24
        Funkcija prima podatak tipa int i vraca podatak tipa int */

int intervalH (int val)
{
        if ( val > 24)
        {
                val = 1;
        }
        if (val < 0)
        {
                val = 24;
        }
        return val;
}

/*      Funkcija koja regulira da ce primljeni podatak biti
        u intervalu 0 do 59
        Funkcija prima podatak tipa int i vraca podatak tipa int */

int intervalM (int val)
{
        if ( val > 59)
        {
                val = 0;
        }
        if (val < 0)
        {
                val = 59;
        }
        return val;
}

/*      Funkcija koja prati dali je proslo odredjeno
         vrijeme ovisno o varijabli gateIntervalu
        Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void gateTimer(void)
{
        RTC_Read_Clock(0);
        int currentTime = BCDToDecimal(minute);
        int currentTimeSec = BCDToDecimal(second);
        if ( currentTime < pressedEvent)
        {
                currentTime = currentTime + pressedEvent;
        }
        if (currentTime-pressedEvent >= gateInterval)
        {
                if (currentTimeSec >= pressedEventSec)
                {
                        gateFlag = false;
                }
        }
}

/*      Funkcija koja provjeri dali su vrata otvorena,
```

```
	ako su otvorena zapamti vrijeme i postavi zastavicu
	Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void gateEvent(void)
{
	if(button_state(gate))
	{
		pressedEvent = pmin;
		pressedEventSec = psec;
		gateFlag = true;
	}
}

/*	Funkcija koja salje s atmega16 na esp8266 preko uarta
	AT komandu za spajanje na stranicu
	Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void esp_site(void)
{
	char _atCommand[60];
	sprintf(_atCommand, "AT+CIPSTART=\"TCP\",\"%s\",%s", DOMAIN, PORT);
	USART_SendString(_atCommand);
	USART_SendString("\r\n");
	_delay_ms(1500);
}

/*	Funkcija koja salje s atmega16 na esp8266 preko uarta
	AT komandu koja ce rec kolko dugacku poruku saljemo
	i nakon toga saljemo podatak GET ..... za zapis podatka na stranici
	Funkcija prima podatake i svi su tipa uint8_t
	field - koje polje na stranici thingspeak se upisuje podatak
	d_num - kolko podataka ce se zapisati u polje
	data1 - podatak 1 koji se upisuje u polje
	data2 - podatak 2 koji se upisuje u polje
	Funkcija nevraca nikakav podatak */

void esp_publish(uint8_t field, uint8_t d_num, uint8_t data1, uint8_t data2)
{
	char _atCommand[60];
	char get_buffer[150];
	if (d_num == 1)
	{
		sprintf(get_buffer, "GET
https://api.thingspeak.com/update?api_key=%s&field%d=%d",API_WRITE_KEY,field,data1);
	}
	else if (d_num ==2)
	{
		sprintf(get_buffer, "GET
https://api.thingspeak.com/update?api_key=%s&field%d=%d%d",API_WRITE_KEY,field,data1,d
ata2);
	}

	sprintf(_atCommand, "AT+CIPSEND=%d", (strlen(get_buffer)+2));
	USART_SendString(_atCommand);
	USART_SendString("\r\n");
	_delay_ms(1500);
	USART_SendString(get_buffer);
	USART_SendString("\r\n");
	_delay_ms(1500);
}

/*	Funkcija koja salje s atmega16 na esp8266 preko uarta
```

```
	AT komandu koja ce rec kolko dugacku poruku saljemo
	i nakon toga saljemo podatak GET ..... za citanje podatka sa stranice
	Funkcija prima podatak tipa uint8_t
	field - koje polje na stranici thingspeak se upisuje podatak
	Funkcija nevraca nikakav podatak */

void esp_local_update(uint8_t field)
{
	char _atCommand[60];
	char get_buffer[150];
	sprintf(get_buffer, "GET
https://api.thingspeak.com/channels/%s/fields/%d/last.txt",CHANNEL_ID,field);
	sprintf(_atCommand, "AT+CIPSEND=%d", (strlen(get_buffer)+2));

	USART_SendString(_atCommand);
	USART_SendString("\r\n");
	_delay_ms(1500);
	USART_SendString(get_buffer);
	USART_SendString("\r\n");

}

/*	Funkcija koja ovisno o primljenoj poruci s esp8266 preko uarta,
	(primljena poruka se nalazi u uart_poruka),
	obnavlja vrijednosti lokalnih varijabli ili postavlja flag za wifi,
	Funkcija prima podatak tipa uint8_t
	field - koje polje na stranici thingspeak se upisuje podatak
	Funkcija nevraca nikakav podatak */

void uart_korisni_dio(uint8_t field)
{
	if (strstr(uart_poruka,"CLO")!=NULL)
	{
		ret = strstr(uart_poruka,"+IPD");  // uzmi adresu gdje pocinje "+IPD" u
pristigloj poruci
		int vrijednost=atoi(ret+7);		//korisni podatak se nalazi na
+7 adresi od ret +IPD

		//provjeravamo gdje moramo upisati podatak
		if (field==1)
		{
			hourONtime=vrijednost/100;
			minuteONtime=vrijednost%100;
		}
		else if (field==2)
		{
			hourOFFtime=vrijednost/100;
			minuteOFFtime=vrijednost%100;
		}
		else if (field==3)
		{
			gateInterval=vrijednost;
		}
		else if (field==4)
		{
			gateOFFtimeH=vrijednost/100;
			gateOFFtimeM=vrijednost%100;
		}
		//ocistimo uart_poruka i ret
		memset(uart_poruka,0,150);
		ret =0;
	}
```

```
	else if (strstr(uart_poruka,"STATUS:")!=NULL)
	{
		ret = strstr(uart_poruka,"STATUS:");
		int vrijednost = atoi(ret+7);

		if (vrijednost==5)
			wifi_stanje=false;
		else
			wifi_stanje=true;
		//ocistimo uart_poruka i ret
		memset(uart_poruka,0,150);
		ret =0;
	}
	else if (strstr(uart_poruka,"WIFI CONNECTED")!=NULL)
	{
		wifi_stanje=true;
		//ocistimo uart_poruka i ret
		memset(uart_poruka,0,150);
		ret =0;
	}
	else if (strstr(uart_poruka,"FAIL")!=NULL)
	{
		wifi_stanje=false;
		//ocistimo uart_poruka i ret
		memset(uart_poruka,0,150);
		ret =0;
	}
}

/*	Funkcija koja prima poruku s esp8266 preko uarta,
	(primljena poruka se nalazi u uart_poruka),
	Funkcija prima podatak tipa uint8_t i char*
	field - koje polje na stranici thingspeak se upisuje podatak
	ocekivani_odgovor - podatak koji ocekujemo ukoliko smo uspjesno poslali poruku
	neocekivani_odgovor + podatak koji ocekujemo ukoliko smo neuspjesno poslali
poruku
	Funkcija nevraca nikakav podatak */

void uart_primi(uint8_t field,char* ocekivani_odgovor,char* neocekivani_odgovor)
{
	uint8_t i=0;
	//primaj poruku sve dok nedodjes do ocekivani_odgovor ili neocekivani_odgovor
	do
	{
		uart_poruka[i]=USART_Recieve();
		i++;
		if (i==150)
		{
			i=0;
		}
	}while
((strstr(uart_poruka,ocekivani_odgovor)==NULL)&&(strstr(uart_poruka,neocekivani_odgovo
r)==NULL));


	if (strstr(uart_poruka,"ERROR")!=NULL)
	{
		memset(uart_poruka,0,150);
		ret =0;
	}
	uart_korisni_dio(field);
}
```

```
/* Funkcija koja salje s atmega16 na esp8266 preko uarta
       AT komandu za spajanje na wifi mrezu
       Funkcija neprima nikakav podatak i nevraca nikakav podatak */

void esp_wifi(void)
{
       char _atCommand[60];
       sprintf(_atCommand, "AT+CWJAP=\"%s\",\"%s\"", SSID, PASSWORD);
       USART_SendString(_atCommand);
       USART_SendString("\r\n");
       uart_primi(0,"WIFI CONNECTED","FAIL");
       _delay_ms(2000);
}



int main(void)
{
       //int b=0;
//***** OUTPUT ********
       DDRD = 0b11111100;   // ukljuci sve pinove za port D
       PORTD = 0b11111100;  //pocetna vrijednost outputa 1
       DDRC = 0b11000000;   // ukljuci sve pinove za port D
       PORTC = 0b11000000;
//***** INPUT ********
       DDRB = 0x00;         // postavit  pinove na input na portu B
       PORTB = 0b01111111;  // odabrati koristenje zeljenog moda
       SFIOR &= ~(1<<PUD);  // odabrati koristenje zeljenog moda

       I2C_Init();                        /* Initialize I2C */
       lcd_init(LCD_DISP_ON);      /* Initialize LCD16x2 */
       USART_Init(115200);

       //RTC_Clock_Write(0x20, 0x01, 0x00, hour_24); //Zapisuje sat, minute, sekunde u
tocno ovom formatu
       //RTC_Calendar_Write(0x1, 0x30, 0x8, 0x21);     // Zapisuje redni broj dana u
tjednu, dan u mjesecu, mjesec i godinu

       _delay_ms(1000);
       USART_SendString("AT+CIPSTATUS\r\n");
       uart_primi(1,"OK","ERROR");
       if (wifi_stanje==false)
       {
              esp_wifi();
       }

    while(1)
    {
              RTC_Read_Clock(0);
              psat =BCDToDecimal(hour) ;
              pmin = BCDToDecimal(minute);
              psec = BCDToDecimal(second);

              switch (screenPicker)       // switch case pomocu kojeg biramo na kojem
smo ekranu
              {
//***** home screen ********
              case 0:

                     screen0();
```

```
if (psec==30)
{
	USART_SendString("AT+CIPSTATUS\r\n");
	uart_primi(1,"OK","ERROR");
	lcd_gotoxy(15,0);
	lcd_puts("*");
	if (wifi_stanje==false)
		esp_wifi();

	if (wifi_stanje==true)
	{

		esp_site();
		esp_local_update(1);
		uart_primi(1,"CLO","ERROR");
		esp_site();
		esp_local_update(2);
		uart_primi(2,"CLO","ERROR");
		esp_site();
		esp_local_update(3);
		uart_primi(3,"CLO","ERROR");
		esp_site();
		esp_local_update(4);
		uart_primi(4,"CLO","ERROR");
		lcd_gotoxy(15,0);
		lcd_puts(" ");
	}
}

if (button_state(edit))
{
	int postaviH = psat;
	int postaviM = pmin;

	do
	{
		switch (editMenu)
		{
			case 0:

			lcd_gotoxy(4,0);
			lcd_puts(">");
			screen01(postaviH,postaviM);
			if (button_state(plus))
			{
				postaviH++;
				postaviH = intervalH(postaviH);
				screen01(postaviH,postaviM);
				_delay_ms(200);
			}
			if (button_state(minus))
			{
				postaviH--;
				postaviH = intervalH(postaviH);
				screen01(postaviH,postaviM);
				_delay_ms(200);
			}
			if (button_state(next))
			{
				editMenu++;
				_delay_ms(200);
```

```
                                        lcd_clrscr();

                                }
                                break;

                                case 1:
                                lcd_gotoxy(10,0);
                                lcd_puts("<");
                                screen01(postaviH,postaviM);
                                if (button_state(plus))
                                {
                                        postaviM++;
                                        postaviM = intervalM(postaviM);
                                        screen01(postaviH,postaviM);
                                        _delay_ms(200);
                                }
                                if (button_state(minus))
                                {
                                        postaviM--;
                                        postaviM = intervalM(postaviM);
                                        screen01(postaviH,postaviM);
                                        _delay_ms(200);
                                }
                                if (button_state(next))
                                {
                                        editMenu++;
                                        _delay_ms(200);
                                        lcd_clrscr();
                                }
                                if (button_state(prev))
                                {
                                        editMenu--;
                                        _delay_ms(200);
                                        lcd_clrscr();

                                }
                                break;
                        }
                } while (button_state(set)!=1);
                if (postaviH > 9 && postaviH < 20)
                        postaviH = postaviH + 6;
                else if(postaviH >= 20)
                        postaviH = postaviH + 12;

                if (postaviM > 9 && postaviM < 20)
                        postaviM = postaviM + 6;
                else if(postaviM >= 20 && postaviM < 30)
                        postaviM = postaviM + 12;
                else if(postaviM >= 30 && postaviM < 40)
                        postaviM = postaviM + 18;
                else if(postaviM >= 40 && postaviM < 50)
                        postaviM = postaviM + 24;
                else if(postaviM >= 50)
                        postaviM = postaviM + 30;

                RTC_Clock_Write(postaviH, postaviM, 0x00, hour_24);
                lcd_clrscr();// ocisti ekran kad izadjes iz edit
moda
        }
```

```
                    if (button_state(next))           // provjera dali je pritisnuta
tipka next
                    {
                           screenPicker++;                  // povecamo varijablu
kako bi usli u case 1 od switch case (screen 1)
                           lcd_clrscr();              // obrisemo sve s ekrana
                           _delay_ms(200);                  // pricekamo 200 ms kako
nebi jedan pritisak registrirao vise puta
                    }
                break;
//***** screen 1 **********
              case 1:
                    screen1();
                    if (button_state(edit))                 // provjera dali je
stisnuta edit tipka, ako je ulazimo u edit mode
                    {
                           do                                       // zadrzavamo se
u edit modu sve dok se nepritisne tipka set
                           {
                                  switch (editMenu)    // switch case s kojim
odredujemo koju varijablu zelimo editirati
                                  {
                                         case 0:                           //
editiranje sati kada se pali svjetlo

                                                lcd_gotoxy(0,1);
                                                lcd_puts(">");              // pomocu
ovog korisniku dajemo do znanja da je u edit modu i koju varijablu trenutno editira
                                                screen1();
                                                if (button_state(plus))           //
ako se stisne ili drzi tipka plus povecavaj brojcanik
                                                {
                                                       hourONtime++;
                                                       hourONtime =
intervalH(hourONtime);
                                                       screen1();
                                                       _delay_ms(200);             //
omogucuje da pritisak registrira samo jednom a drzanje vise puta
                                                }
                                                if (button_state(minus))// ako se
stisne ili drzi tipka minus smanjuj brojcanik
                                                {
                                                       hourONtime--;
                                                       hourONtime =
intervalH(hourONtime);
                                                       screen1();
                                                       _delay_ms(200);
                                                }
                                                if (button_state(next))     // ako se
stisne tipka next prelazimo na sljedecu varijablu koju cemo editirat
                                                {
                                                       editMenu++;
                                                       _delay_ms(200);
                                                       lcd_clrscr();

                                                }
                                         break;

                                         case 1:

                                                lcd_gotoxy(6,1);
                                                lcd_puts("<");
```

```
                                        screen1();

                                        if (button_state(plus))
                                        {
                                                minuteONtime++;
                                                minuteONtime =
intervalM(minuteONtime);
                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(minus))
                                        {
                                                minuteONtime--;
                                                minuteONtime =
intervalM(minuteONtime);
                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(next))
                                        {
                                                editMenu++;
                                                _delay_ms(200);
                                                lcd_clrscr();
                                        }
                                        if (button_state(prev))
                                        {
                                                editMenu--;
                                                _delay_ms(200);
                                                lcd_clrscr();

                                        }
                                break;

                                case 2:
                                        lcd_gotoxy(9,1);
                                        lcd_puts(">");
                                        screen1();
                                        if (button_state(plus))
                                        {
                                                hourOFFtime++;
                                                hourOFFtime =
intervalH(hourOFFtime);
                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(minus))
                                        {
                                                hourOFFtime--;
                                                hourOFFtime =
intervalH(hourOFFtime);
                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(next))
                                        {
                                                editMenu++;
                                                _delay_ms(200);
                                                lcd_clrscr();
                                        }
                                        if (button_state(prev))
                                        {
                                                editMenu--;
```

```
                                                _delay_ms(200);
                                                lcd_clrscr();
                                        }
                                break;

                                case 3:
                                        lcd_gotoxy(15,1);
                                        lcd_puts("<");
                                        screen1();
                                        if (button_state(plus))
                                        {
                                                minuteOFFtime++;
                                                minuteOFFtime =
intervalM(minuteOFFtime);

                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(minus))
                                        {
                                                minuteOFFtime--;
                                                minuteOFFtime =
intervalM(minuteOFFtime);

                                                screen1();
                                                _delay_ms(200);
                                        }
                                        if (button_state(prev))
                                        {
                                                editMenu--;
                                                _delay_ms(200);
                                                lcd_clrscr();
                                        }
                                break;
                                }
                        } while (button_state(set)!=1);          // zadrzavamo se
u edit modu sve dok se nepritisne tipka set
                        if (wifi_stanje == true)
                        {
                                lcd_clrscr();
                                lcd_gotoxy(4,0);
                                lcd_puts("UPDATING");
                                esp_site();
                                esp_publish(1,2,hourONtime,minuteONtime);
                                _delay_ms(15500);
                                esp_site();
                                esp_publish(2,2,hourOFFtime,minuteOFFtime);
                        }
                        lcd_clrscr();               // ocisti ekran kad izadjes iz
edit moda
                }
                if (button_state(next))           // provjera dali pritisnuta
tipka next da bi osli na sljedeci ekran
                {
                        screenPicker++;
                        lcd_clrscr();
                        _delay_ms(200);
                }

                if (button_state(prev))           // provjera dali pritisnuta
tipka prev da bi osli na prosli ekran
                {
                        screenPicker--;
                        lcd_clrscr();
```

```
                        _delay_ms(200);
                }
            break;
//******* screen 2 ***********
            case 2:
                screen2();
                editMenu=0;                                 // postavimo editMenu na
0 kako bi krenili editiranje od prve varijable
                if (button_state(edit))
                {
                    do
                    {
                        switch (editMenu)
                        {
                            case 0:

                                lcd_gotoxy(6,0);
                                lcd_puts(">");
                                screen2();
                                if (button_state(plus))
                                {
                                    gateInterval++;
                                    gateInterval =
intervalM(gateInterval);

                                    screen2();
                                    _delay_ms(200);
                                }
                                if (button_state(minus))
                                {
                                    gateInterval--;
                                    gateInterval =
intervalM(gateInterval);

                                    screen2();
                                    _delay_ms(200);
                                }
                                if (button_state(next))
                                {
                                    editMenu++;
                                    _delay_ms(200);
                                    lcd_clrscr();

                                }
                            break;

                            case 1:
                                lcd_gotoxy(3,1);
                                lcd_puts(">");
                                screen2();
                                if (button_state(plus))
                                {
                                    gateOFFtimeH++;
                                    gateOFFtimeH =
intervalH(gateOFFtimeH);

                                    screen2();
                                    _delay_ms(200);
                                }
                                if (button_state(minus))
                                {
                                    gateOFFtimeH--;
                                    gateOFFtimeH =
intervalH(gateOFFtimeH);

                                    screen2();
```

```
                                        _delay_ms(200);
                                }
                                if (button_state(next))
                                {
                                        editMenu++;
                                        _delay_ms(200);
                                        lcd_clrscr();
                                }
                                if (button_state(prev))
                                {
                                        editMenu--;
                                        _delay_ms(200);
                                        lcd_clrscr();

                                }
                        break;

                        case 2:
                                lcd_gotoxy(10,1);
                                lcd_puts("<");
                                screen2();
                                if (button_state(plus))
                                {
                                        gateOFFtimeM++;
                                        gateOFFtimeM =
intervalM(gateOFFtimeM);
                                        screen2();
                                        _delay_ms(200);
                                }
                                if (button_state(minus))
                                {
                                        gateOFFtimeM--;
                                        gateOFFtimeM =
intervalM(gateOFFtimeM);
                                        screen2();
                                        _delay_ms(200);
                                }
                                if (button_state(prev))
                                {
                                        editMenu--;
                                        _delay_ms(200);
                                        lcd_clrscr();
                                }
                        break;
                }
        } while (button_state(set)!=1);

        if (wifi_stanje == true)
        {
                lcd_clrscr();
                lcd_gotoxy(4,0);
                lcd_puts("UPDATING");
                esp_site();
                esp_publish(3,1,gateInterval,0);
                _delay_ms(15500);
                esp_site();
                esp_publish(4,2,gateOFFtimeH,gateOFFtimeM);
        }
        lcd_clrscr();// ocisti ekran kad izadjes iz edit moda
}
```

```
                    if (button_state(prev))                 // na zadnjem ekranu
ocekujemo samo pritisak tipke za odlazak na prosli ekran
                    {
                           screenPicker = screenPicker-1;
                           lcd_clrscr();
                           _delay_ms(200);
                    }

             break;
             }

             //************************************************
             // paljenje i gasenje
             //************************************************

             if (gateFlag == false)
             // ovaj dio koda ce se izvrsavat samo ako su vrata zatvorena (nisu bila
otvorena u odredjenom intervalu)
             {
                    if ( (psat > hourONtime) && (psat < hourOFFtime) )     // ovaj
dio gleda kada su sati > <
                           light = true;
                    else if (psat == hourONtime && pmin >= minuteONtime) // ovaj dio
gleda za tocno odredeni sat i minute ON TIME
                           light = true;
                    else if (psat == hourOFFtime && pmin < minuteOFFtime)  // ovaj
dio gleda za tocno odredeni sat i minute OFF TIME
                           light = true;
                    else
                           light = false;
             }
//******** gate stuff *********************************************
             if ((psat > hourOFFtime) && (psat < gateOFFtimeH))            // ovo ce
se izvrsavat od trenutka gasenja svijetla do gate off time
                    gateEvent();
             else if ((psat == hourOFFtime) && (pmin >= minuteOFFtime))
                    gateEvent();
             else if (psat == gateOFFtimeH && pmin <= gateOFFtimeM )
                    gateEvent();
             else if (gateOFFtimeH < hourOFFtime)            // ako je vrijeme
gasenja od vrata manje od OFF time
             {
       // podjeli na dva intervala i provjeri u kojem je
                    if (psat > hourOFFtime && psat <= 24)    // od gasenja do ponoci
                           gateEvent();
                    if (psat < gateOFFtimeH)                 // od 01 do gasenja od
vrata
                           gateEvent();
             }

             if (gateFlag == true)             // ako su vrata otvorena/bila otvorena
pocni odbrojavanje
                    gateTimer();
             if (gateFlag == true)             // ako su vrata otvorena/bila otvorena
(i unutar intervala od timera) upali sv
                    light = true;

             if (light)                  //upali svjetlo ovisno o zastavici za light
                    turnON();
             else
                    turnOFF();          //ugasi svjetlo ovisno o zastavici za light
```

```
    }
}
```

Uart.c

```
/*
 * Uart.c
 *
 * Created: 24/08/2021 15:55:39
 *  Author: vkrus
 */

#include "Uart.h"

void USART_Init(unsigned long BAUDRATE)         // uart inicijalizacija
{
       uint16_t baudPrescaler;
       baudPrescaler = (F_CPU/(16UL*BAUDRATE))-1;

       UCSRA &=~(1 << U2X);

       UCSRB |= (1 << RXEN) | (1 << TXEN);                          // ukljuci uart
tx rx
       UCSRC |= (1 << URSEL)| (1 << UCSZ0) | (1 << UCSZ1);    // USCRC za 8 bit data i
1 stop bit
       UBRRL = baudPrescaler;                          // ucitaj UBRRL sa doljnim 8
bitova od prescale vrijednosti
       UBRRH = (baudPrescaler >> 8);            // ucitaj UBRRH sa gornjim 8 bitova od
prescale vrijednosti
}

char USART_Recieve()                      // Funkcija za primanje podataka
{
       while (!(UCSRA & (1 << RXC)));           // cekaj dok se neprimi novi podatak
       return(UDR);                                    //vrati dobiveni podatak
}

void USART_Send(char data)                // funkcija za slanje podataka
{
       while (!(UCSRA & (1<<UDRE)));
       UDR = data;
}

void USART_SendString(char *str)         // funkcija za slanje stringa
{
       int i=0;
       while (str[i]!=0)
       {
              USART_Send(str[i]);               //salji znak do null
              i++;
       }
}
```

Uart.h

```
/*
 * Uart.h
 *
 * Created: 24/08/2021 15:54:12
 *  Author: vkrus
 */


#ifndef UART_H_
#define UART_H_

#define F_CPU 11059200UL
#include <avr/io.h>
#include <math.h>


void USART_Init(unsigned long);          // uart inicijalizacija
char USART_Recieve();                          // Funkcija za primanje podataka
void USART_Send(char);                         // funkcija za slanje podataka
void USART_SendString(char*);            // funkcija za slanje stringa

#endif /* UART_H_ */
```