Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za automatiku i elektroniku
Praktikum automatike

Sistem za naručivanje hrane u restoranu

PROJEKTNI ZADATAK

Prof. dr. Samim Konjicija, dipl. el. ing.
Mr. Nedim Osmić, dipl. el. ing.

Almir Bešić
Semir Berković
Mirzet Brkić

Sarajevo, januar – juni, 2016.g.

Sadržaj

1. Uvod

Većina restorana u našoj zemlji ima stroge kriterije kada je riječ o usluzi mušterija. Stremi se ka evropskim standardima i pravilima, ali napredak je znatno spor. Najveći zaostatak u ovim djelatnostima je primjetan u manjim fast food restoranima gdje su najmnogobrojnije mušterije studenti. Naime, narudžba i preuzimanje jela predstavlja komplikovan proces koji zahtjeva dobar sluh i punu pažnju. Hrana se može preuzeti samo ako ste čuli zaposlenika restorana kako proziva vašu narudžbu. Problem postaje veći pred ispitne rokove kada na takvim mjestima u istom trenutku bude 50 studenata i od tog broja 10 njih napravi istu narudžbu. Možete samo zamisliti razočaranje kada ugledate da je neka druga osoba preuzela vaš sendvič koji ste čekali 20 minuta da se napravi. Tada nastaju svađe sa uposlenicima i drugim mušterijama, jer ne postoji informacija koja povezuje narudžbu i naručioca.

Da bi izbjegli sve pomenute neugodne situacije i sačuvali glasnice uposlenika (jer smo i sami bili žrtve ovog sistema), raspravljali smo o rješenju problema. Tako je nastala ideja o sistemu naručivanja hrane.

Sistem se sastoji od centralnog transmitera i više manjih tzv. pejdžera. Transmiter evidentira narudžbe i prenosi ih na računar u aplikaciju gdje se pohranjuju određeni podaci (elementi narudžbe, broj pejdžera, itd.). Nakon evidencije, mušteriji se predaje pejdžer pomoću kojeg će biti obavješten kada može preuzeti narudžbu.

2. Sistem za naručivanje hrane u restoranu

Kao što je rečeno, sistem se sastoji od dvije osnovne komponente, transmiter i pejdžer(i). Transmiter predstavlja bazni (master) uređaj koji upravlja ostalim pejdžerima (slave) koristeći se RF modulima. Dizajniran je tako da radi sa mikrokontrolerom PIC16F1939. Napaja se pomoću USB-a, koji je spojen na PC na kojem se nalazi odgovarajuća aplikacija za upravljanje sistemom. Koristeći se GUI-em vrlo jednostavno se vrši prozivanje pejdžera i evidencija narudžbi, što je i dovoljno uposlenicima za rad.

Pejdžeri su uređaji koji se predaju mušterijama nakon narudžbe i vraćaju prilikom preuzimanja iste. Imaju ugrađen resiver preko kojeg primaju naredbe transmitera. Napaja se LiIon baterijom i za ove module je napravljen punjač. Korišten je mikrokontroler PICF1877A, koji je za nijansu slabiji od PIC16F1939, radi ekonomičnosti. Pejdžer se može nalaziti u četiri stanja:

- *Čekanje* – pejdžer miruje.
- *Prozivanje* – narudžba je gotova, mušterija biva obavješten zvučnim i vizuelnim signalima.
- *Izvan opsega* – ukoliko neko pokuša napustiti radni opseg sa pejdžerom, bit će obavješten određenim zvučnim i vizuelnim signalima.
- *Prazna baterija* – ukoliko se baterija isprazni potrebno je staviti pejdžer na punjenje. Prikladna vizuelna upozorenja su isprogramirana.

2.1. Centralni transmiter

2.1.1. Komponente i interfejsi

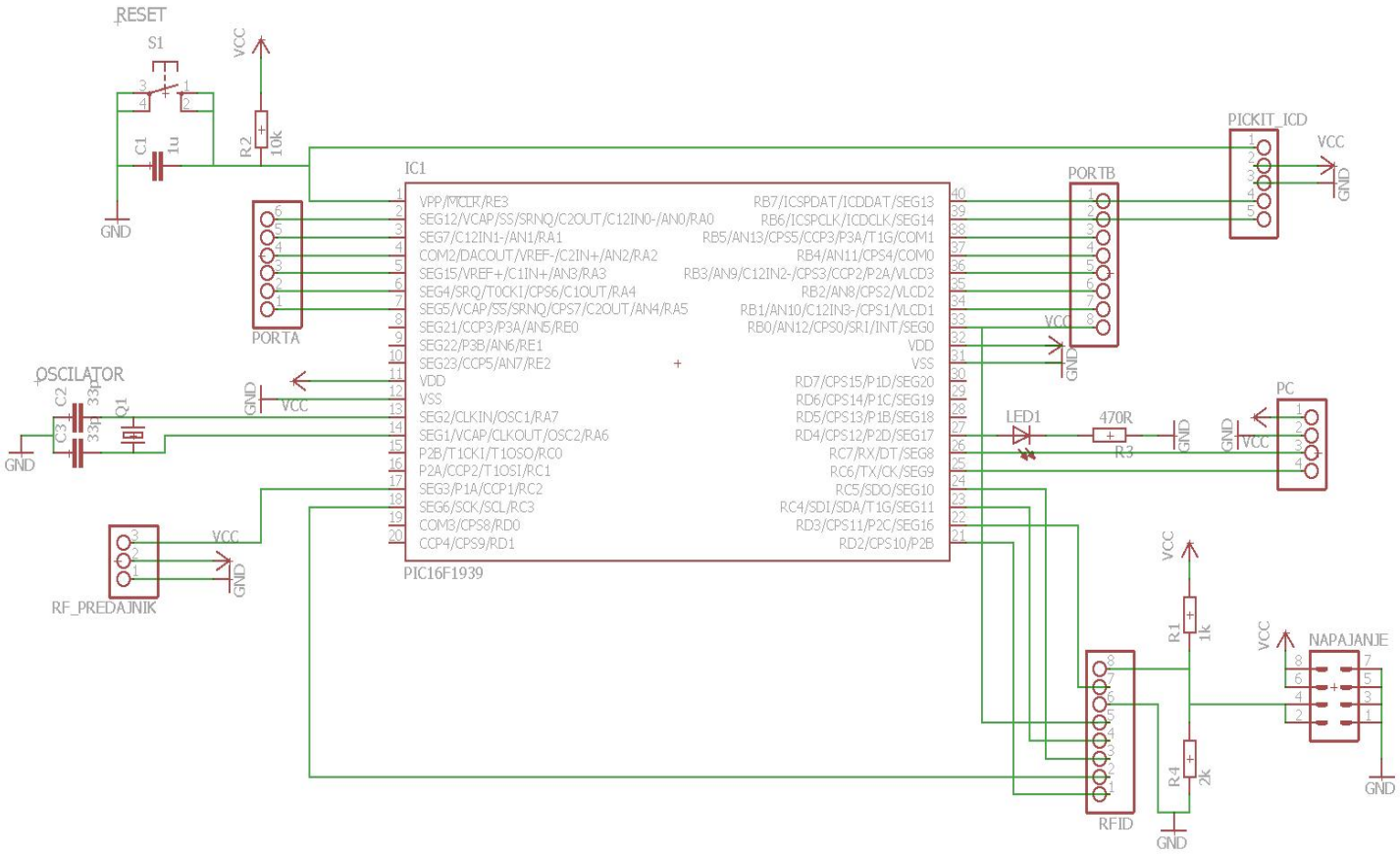
- 1 x mikrokontroler PIC16F1939
- 1 x RF 433MHz predajnik
- 1 x kristalni oscilator 8 MHz
- 4 x otpornik (1 x 470 Ω, 1 x 10 kΩ, 1 x 1 kΩ, 1 x 2 kΩ)
- 3 x kondenzator (1 x 1 μF, 2 x 33 pF)
- 1 x LED
- 1 x reset dugme
- 1 x PICKIT header za programiranje mikrokontrolera
- pin headeri različitih veličina (muški i ženski)

- Programiranje PIC mikrokontrolera se izvodi pomoću PICKIT uređaja. U tu svrhu je izvedeno 5 pin-ova koji su potrebni za priključenje samog uređaja (RE3, 5V, GND, RB7 i RB6).
- Kompletni portovi A i B su izvedeni kao ženski pin header-i radi lakšeg pristupa u slučaju da se pojavi potreba za njihovim korištenjem.
- [nije programski realizovano] Na portu RD4 spojena je LED dioda koja indicira da li je moguće skenirati RF tag koji se nalaze na pejdžerima (u slučaju naručivanja ili preuzimanja narudžbe).

- 8 MHz kristalni oscilator je spojen na pin OSC1 i OSC2 preko 2 kondenzatora radi generisanja radnog takta.
- 8 ženskih pin-ova je izvedeno za RFID. Napravljen je naponski djelitelj sa otpornicima R1 i R4 koji ostvaruje napon od 3.3 V na pinu za napajanje RFID uređaja (3.3V, RD3, GND, RB0, RC4, RC5, RC3 i RD2).
- 4 ženska pin-a su izvedena za svrhe USB na TTL priključka (5V, GND, RC7 i RC6).
- 3 ženska pin-a su izvedena za priključak transmiter modula RF komunikacije (5V, GND i RC2).
- 8 muških pin-ova izvedenih kao VCC i GND (2 x 3.3V, 2 x 5V i 4 x GND).
- Ugrađeno je reset dugme na pin-u RE3 koji služi za restartovanje cijelog uređaja u slučaju pojavljivanja greške.

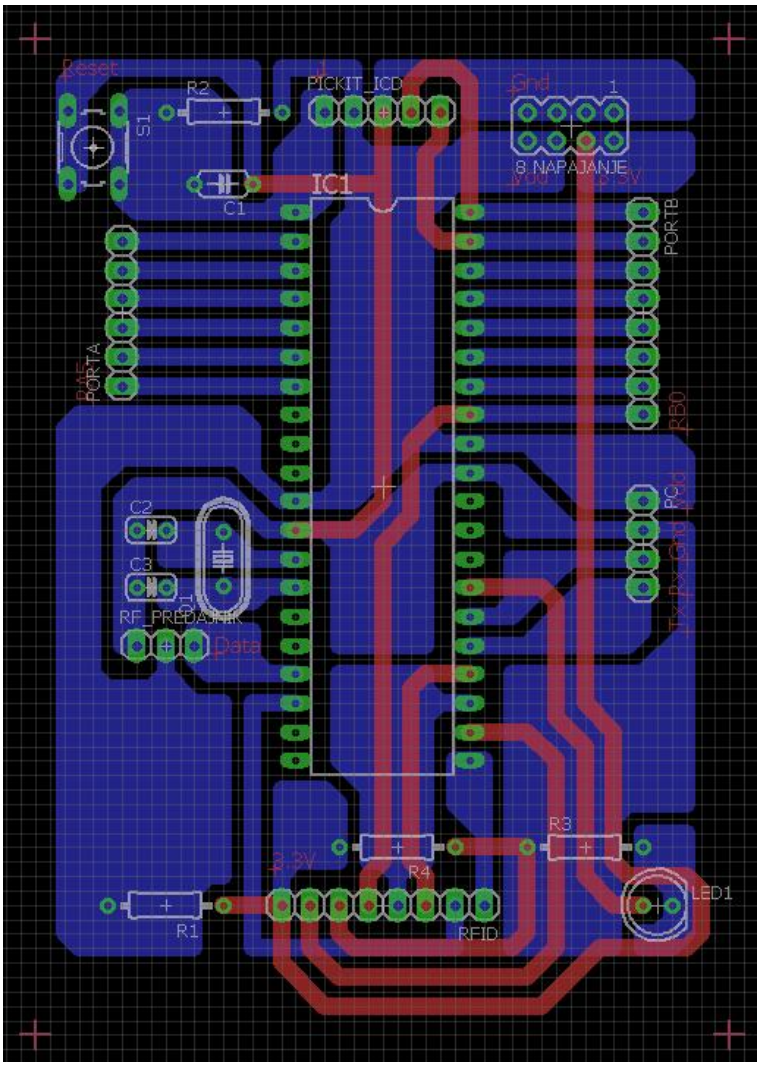
2.1.2. Logička shema

Logičke sheme i PCB dizajn su rađeni korištenjem programskog paketa Eagle.



Slika 2.1. Logička shema centralnog transmitera

2.1.3. PCB dizajn



Slika 2.2. Printed circuirtboard (PCB) dizajn centralnog transmitera

2.2. Pager

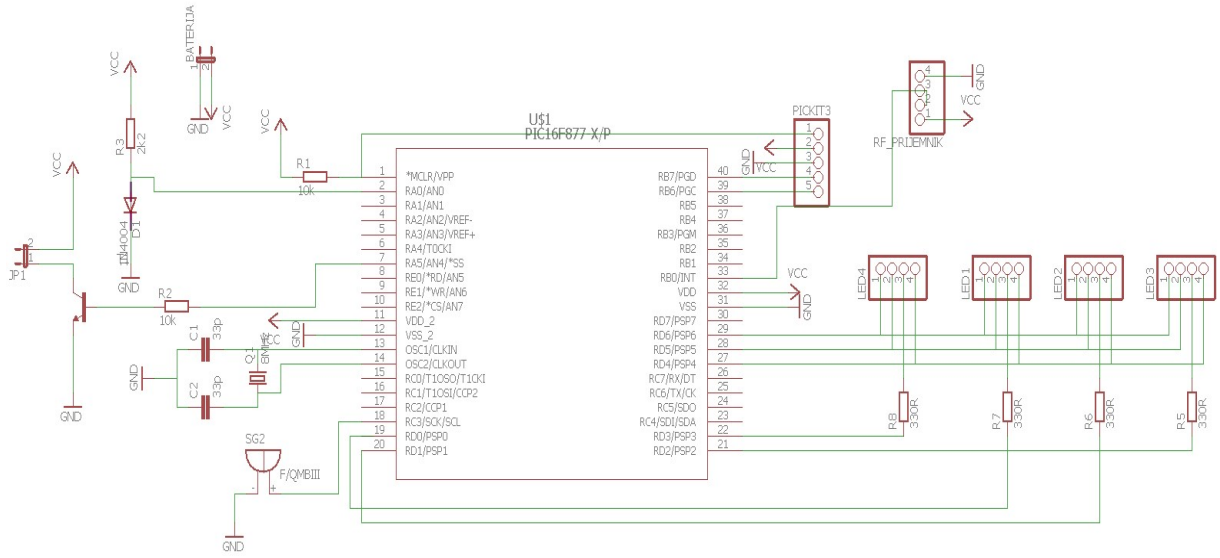
Mušterija nakon završene narudžbe dobija pager, sa jedinstvenim identicikacijskim brojem zapisanim u samom kodu pagera, koji će istu obavijestiti kada narudžba bude gotova. Ukoliko se desi da mušterija nenamjerno (ili namjerno) zaboravi na pager i pokuša napustiti područje unutar kojeg pager radi dolazi do odgovarajućeg zvučnog signal i paljenja RGB dioda koje će emitirati crvenu svjetlost. Ukoliko se desi da je baterija koja napaja pager prazna RGB diode emitiraju crvenu svjetlost. Ukoliko se ne desi ništa od prethodnog i narudžba u međuvremenu bude gotova, centralni transmiter šalje poruku koja jedinstveno identificira pager koji obavještava mušteriju da je narudžba gotova tako što dolazi do odgovarajućeg zvučnog signal, RGB diode emitiraju zelenu svjetlost i dolazi do vibracija pagera. Nakon toga, mušterija prilikom preuzimanja narudžbe vraća pager koji prelazi u stanju mirovanja.

2.2.1.Komponente i interfejsi

- 1 x mikrokontroler PIC16F877A
- 1 x RF 433 MHz prijemnik
- 1 x kristalni oscilator 8 MHz
- 7 x otpornik (4 x 330Ω, 1 x 2.2 kΩ, 2 x 10 kΩ)
- 3 x kondenzator (1 x 1 μF, 2 x 33pF)
- 4 x RGB dioda
- 1 x ispravljačka dioda
- 1 x BC547C tranzistor
- 1 x dvostrana ploča
- 1 x buzzer
- 1 x motor za vibraciju
- 1 x PICKIT header za programiranje mikrokontrolera
- pin headeri različitih veličina (muški i ženski)
- baterijsko napajanje

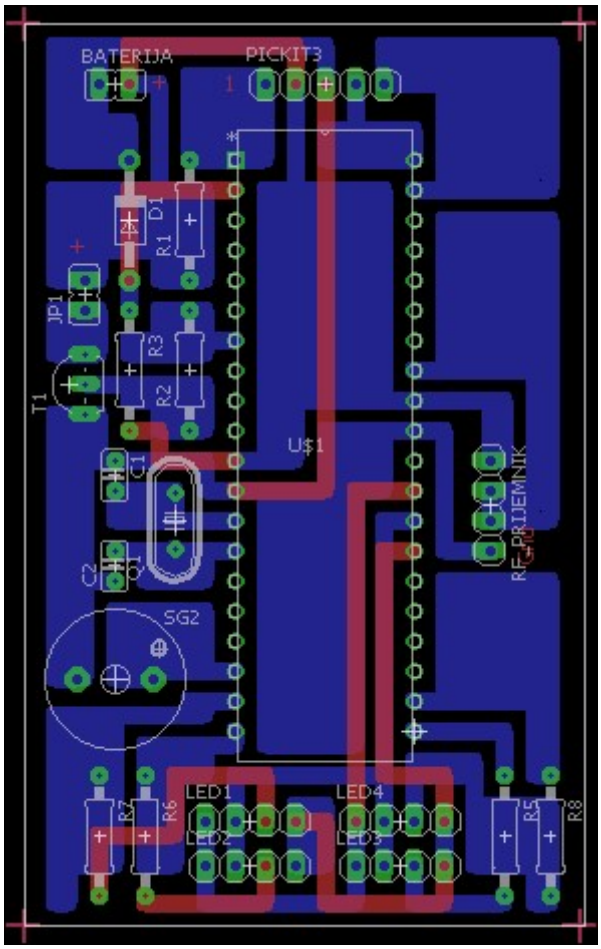
- Programiranje PIC mikrokontrolera se izvodi pomoću PICKIT uređaja. U tu svrhu je izvedeno 5 pin-ova koji su potrebni za priključenje samog uređaja (RE3, 5V, GND, RB7 i RB6).
- 4 pina su izvedena za RF prijemnik (GND, slobodan pin, RB0, 5V).
- Na pinovima RD4, RD5 i RD6 su povezane R, G i B komponente dioda.
- Na pinovima RD0, RD1, RD2 i RD3 su povezane katode RGB dioda.
- 8 MHz kristalni oscilator je spojen na pinove OSC1 i OSC2 preko 2 kondenzatora radi generisanja radnog takta.
- Na pinu RA5 se nalazi motor za vibraciju, pin RA0 je spojen na napon baterije dok je na pinu RC3 buzzer.

2.2.2. Logička shema



Slika 2.2.1 Logička shema pagera

2.2.3. PCB dizajn



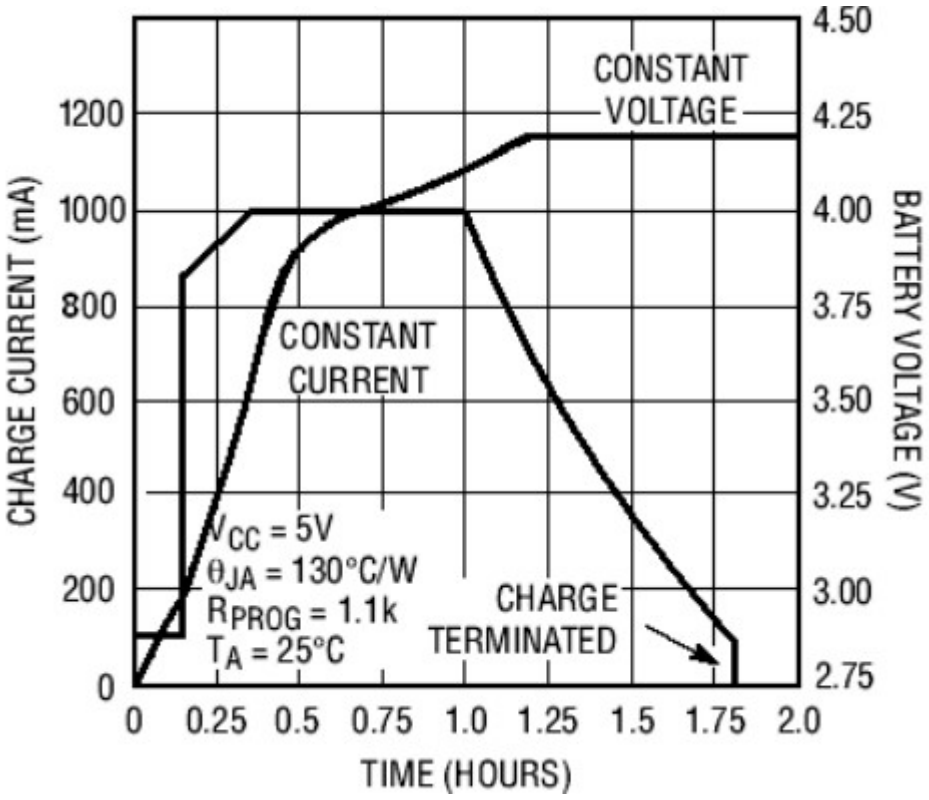
Slika 2.2.2 Printed circuirtboard (PCB) dizajn pagera

2.3. Baterijsko napajanje

2.3.1. Komponente i interfejs

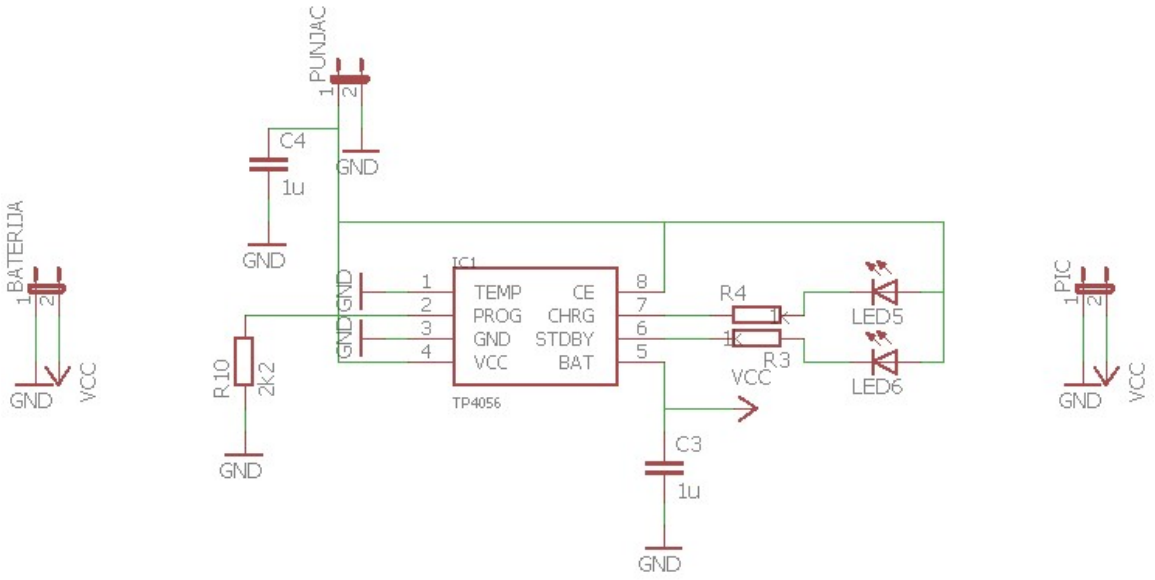
- 1 x TP4056 (1A linearni li-lon punjač sa termoregulacijom)
- 3 x otpornik (2 x 1kΩ, 1 x 2.2 kΩ)
- 2 x kondenzator (2 x 1 μF)
- 2 x LED

Važniji pinovi čipa TP4056:
- pin PROG (pin za postavljanje konstantne struje punjenja i kontrolne struje punjenja): struja punjenja je postavljena spajanjem otpornika R10 ovog pina prema masi.
-pin BAT(pin priključka baterije): pozitivni kraj baterije je spojen na pin BAT. Pin BAT vuče manje od 2 uA struje u modu kada je čip onespesobljen ili sleep modu. BAT pin omogućava punjenje baterije strujom i regulaciju napona na 4.2 V.
-pin STDBY: kada se završi punjenje baterije, pin STDBY sa unutrašnjim prekidačem prelazi na nisko stanje, a inače je ovaj pin u stanju visoke impedanse. LED6 je aktivna kad se dioda napuni.
-pin CHRG: kada se baterija puni, pin CHRG prelazi na nisko stanje unutrašnjim prekidačem, inače je ovaj pin u stanju visoke impedanse. LED5 je aktivna u toku punjenja.



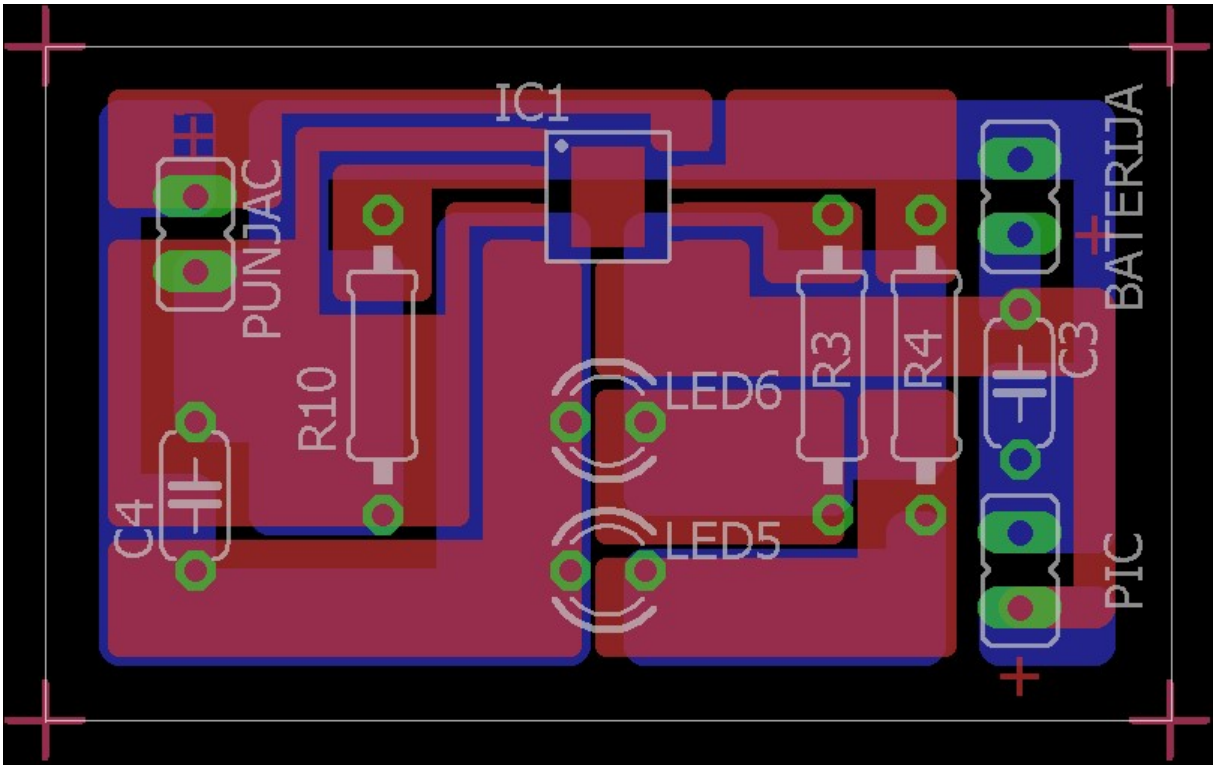
Slika 2.3.1. Ciklus punjenja čipa TP4056 (1000 mAh baterija)

2.3.2. Logička shema



Slika 2.3.2 Logička shema baterijskog napajanja

2.3.3. PCB dizajn



Slika 2.3.3. Printed circuirtboard (PCB) dizajn baterijskog napajanja

3. Zaključak

S obzirom da sistem obavlja svoju osnovnu funkciju, možemo zaključiti da je projekat uspješno realizovan. Kroz rad na ovom projektu još jednom smo potvrdili da su mikrokontroleri dovoljno moćni za realizaciju složenih sistema. Modularna arhitektura sistema sa otvorenim interfejsima omogućava dalju nadogradnju i usavršavanje u pogledu dodavanja novih fukcionalnosti (RFID tagova, dodatne periferije na transmiteru...).

Pored osnovnih osobina za koje je sistem pravljen, dodatne prednosti ovog sistema su: energetska efikasnost, relativno veliki domet (cca 50 m), integracija sa postojećim informacionim sistemom na PC-u, što opravdava uloženo vrijeme i druge resurse u realizaciju ovog projekta. Postojeće znanje stečeno u relizaciji se može, uz određena prilagođenja (korištenje smd komponenti), iskoristiti za relizaciju sistema koji može ići u serijsku industrijsku proizvodnju.

4. Reference

[1] Microchip Technology Inc., *DS40001574C - PIC16(L)F1938/9 28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with LCD Driver,* 2011. - 2013.

[2] Microchip Technology Inc., *DS39582C - PIC16F87XA 28/40/44-Pin Enhanced Flash Microcontrollers,* 2001. - 2013.

[3] Microchip Technology Inc., *DS52053B - MPLAB® XC8 C Compiler User's Guide,* 2012.

[4] Microchip Technology Inc., *DS52116A - PICkit™ 3 In-Circuit Debugger/Programmer User's Guide (For MPLAB® X IDE),* 2013.

[5] NXP Semiconductors, *MFRC522 Standard 3V MIFARE reader solution - Product data sheet,* 17.9.2014.

[5] NanJing Top Power ASIC Corp., *TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8,* 2013.

Svi resursi korišteni na projektu (eagle sheme, pcb desig, kod pagera i transmitera, pc aplikacija i dokumentacija) se nalaze na repozitoriju:
https://github.com/mirzetb/SistemZaNarucivanjeHrane

Dodatak 1. Programski kod

Centralni transmiter

```
/*
 * Embedded solutions and SHIT - Software-Hardware Integration and Testing
 * Elektrotehnicki fakultet Sarajevo
 * Odsjek za automatiku i elektroniku
 * Praktikum automatike
 *
 * Projektni zadatak: Sistem za narucivanje hrane u restoranu
 * File:    Predajnik main.c
 * Authors: Almir Besic
 *          Mirzet Brkic
 *          Semir Berkovic
 */
// PIC16F1939

#include <xc.h>
// CONFIG1
#pragma config FOSC = HS        // Oscillator Selection (HS Oscillator,
High-speed crystal/resonator connected between OSC1 and OSC2 pins)
#pragma config WDTE = OFF       // Watchdog Timer Enable (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable (PWRT disabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select (MCLR/VPP pin
function is MCLR)
#pragma config CP = OFF         // Flash Program Memory Code Protection
(Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Memory Code Protection (Data memory
code protection is disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable (Brown-out Reset
disabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is
disabled. I/O or oscillator function on the CLKOUT pin)
#pragma config IESO = OFF       // Internal/External Switchover
(Internal/External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable (Fail-
Safe Clock Monitor is disabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection
(Write protection off)
#pragma config VCAPEN = OFF     // Voltage Regulator Capacitor Enable (All
VCAP pin functionality is disabled)
#pragma config PLLEN = OFF      // PLL Enable (4x PLL disabled)
#pragma config STVREN = OFF     // Stack Overflow/Underflow Reset Enable
(Stack Overflow or Underflow will not cause a Reset)
#pragma config BORV = LO        // Brown-out Reset Voltage Selection
(Brown-out Reset Voltage (Vbor), low trip point selected.)
#pragma config LVP = OFF        // Low-Voltage Programming Enable (High-
voltage on MCLR/VPP must be used for programming)

// Frekvencija oscilatora
#define _XTAL_FREQ 8000000

// Pomocni makroi za pristup bitima varijable
#define testBit(var, bit) ((var) & (1 << (bit)))
#define setBit(var, bit) ((var) |= (1 << (bit)))
```

```
#define clrBit(var, bit) ((var) &= ~(1 << (bit)))


// Alijasi za komponente/pinove
#define DATA RC2     // Data pin RF predajnika
#define LED RD4      // Anoda LED
#define RST RD3      // Reset pin RFID modula
#define IRQ RB0      // Interrupt pin RFID modula
#define SDI RC4      // SDI/MISO pin RFID modula
#define SDO RC5      // SDO/MOSI pin RFID modula
#define SCK RC3      // SCK pin RFID modula
#define SDA RD2      // SDA pin RFID modula

// Pomocne varijable
char t = 0;
char byteNo = 0;
char rf_data[2];
char pcData;
char send = 0;

void interrupt prekidna_rutina();

// Pomocne funkcije
void rf_send();
void init();
void init_serial();
void init_rf();

void main(void) {
    init();
    while(1){
    }
}

void interrupt prekidna_rutina(){
    if(TMR0IE && TMR0IF){
        TMR0 = 6;
        TMR0IF = 0;
        if(send == 1)
            rf_send();
        else t = 0;
    } else if (RCIE && RCIF) {
        pcData = RCREG;
        if(pcData == '*' || pcData == '#') {
            rf_data[0] = pcData;
            send = 0;
        }
        else {
            rf_data[1] = pcData;
            send = 1;
        }

    }
}

// Inicijalizacija pinova, RF modula, serijske komunikacije i prekida
void init(){
    TRISCbits.TRISC2 = 0;   // RC2 je izlazni [Data pin RF predajnika]
    DATA = 0;

    init_rf();
```

```
    init_serial();
    PEIE = 1;
    RCIE = 1;
    GIE = 1;
}

// Inicijalizacija TIMER0 [osnovni takt rf signala]
void init_rf(){
    T0CS = 0;
    T0SE = 1;
    PSA = 0; // PRESCALER 1:4
    PS2 = 0;
    PS1 = 0;
    PS0 = 1;
    TMR0 = 6;
    TMR0IE = 1;
}

// Funkcija za generisanje naponskog nivoa RF signala (manchester
enkodiranja)
void rf_send(){
    // "zauzimanje" etera - 10 promjena 0->1->0
    if (t == 0) DATA = 1;
    else if (t < 10) DATA = !DATA;
    else if (t < 16) DATA = 1;
    else if (t < 17)  DATA = 0;
    else {
        if (t%2 == 1){
            // Signal podatka
            if (testBit(rf_data[byteNo], (t - 17)/2)) DATA = 1;
            else DATA = 0;
        } else {
            // Diferencijalni signal
            if (testBit(rf_data[byteNo], (t - 17)/2)) DATA = 0;
            else DATA = 1;
        }
    }
    t++;
    // Kraj poruke
    if (t == 34){
        DATA = 0;
        t = 0;
        if(byteNo == 0)
            byteNo = 1;
        else
            byteNo = 0;
    }
}

// Inicijalizacija serijskog modula
void init_serial() {
    SPEN = 1;
    SYNC = 0;
    BRG16 = 0;
    BRGH = 1;
    SPBRGH = 0x00;
    SPBRGL = 51;
    CREN = 1;
    SREN = 1;
    TXEN = 1;
}
```

Pager

```
/*
 * Embedded solutions and SHIT - Software-Hardware Integration and Testing
 * Elektrotehnicki fakultet Sarajevo
 * Odsjek za automatiku i elektroniku
 * Praktikum automatike
 *
 * Projektni zadatak: Sistem za narucivanje hrane u restoranu
 * File:    Pager_v2 main.c
 * Authors: Almir Besic
 *          Mirzet Brkic
 *          Semir Berkovic
 */

// PIC16F877A
#include <xc.h>

// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS
oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR
disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit
Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used
for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit
(Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits
(Write protection off; all program memory may be written to by EECON
control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit
(Code protection off)

// Frekvencija oscilatora
#define _XTAL_FREQ 8000000

// Pomocni makroi za pristup bitima varijable
#define testBit(var, bit) ((var) & (1 << (bit)))
#define setBit(var, bit) ((var) |= (1 << (bit)))
#define clrBit(var, bit) ((var) &= ~(1 << (bit)))

// Alijasi za komponente/pinove
#define VIB RA5     // Vibrator
#define BUZZ RC3    // Buzzer
#define DATA RB0    // Data pin RF prijemnika
#define D1 RD0      // Katoda RGB diode 1
#define D2 RD1      // Katoda RGB diode 2
#define D3 RD2      // Katoda RGB diode 3
#define D4 RD3      // Katoda RGB diode 4
#define BLUE RD6    // B pin RGB dioda
#define GREEN RD5   // G pin RGB dioda
#define RED RD4     // R pin RGB dioda
#define BAT RA0     // Napon baterije
```

```
// Stanje debuggera/komunikacije sa PC-om
#define DEBUG 1

// Identifikacija pagera
char pagerID = 'K';
char opsegID;
char message_header = 0x00;

// Pomocne vrijable
char play = 0;
char loop = 0;
char i = 0;
char brojacLED = 0;
char brojacVIB = 0;
char sound = 0;
char prazna = 0;
// Stanja pagera
enum {CEKANJE, PROZVAN, IZVANOPSEGA} STANJE;
// Stanja baterije
enum {PRAZNA, NIJEPRAZNA} BATERIJA;

// Init metode
void init();
void init_analog();
void init_buzz();
void init_batt();
void init_led();
void init_debug();

// Pmocne metode
void led_show();
void rf_decode();

// Prekidna metoda
void interrupt prekidna_rutina();

void main(void){
    init();
    while(1){
        rf_decode();
    }
}

// Funkcija za dekodiranje RF signala
void rf_decode() {
    char podatak = 0;
    for (i = 0; i < 100; i++){
        __delay_us(20);
        if (DATA == 0) return;
    }

    while(DATA == 1);
    __delay_us(850);

    for(i = 0; i < 8; i++){
        if(DATA == 1)
            setBit(podatak, i);
        else
            clrBit(podatak, i);
        __delay_us(1000);
```

```
    }
    TXREG = podatak;

    if(podatak == '*' || podatak == '#'){
        message_header = podatak;
        opsegID = podatak;
        if(STANJE == IZVANOPSEGA){
            STANJE = CEKANJE;
            RED = 0;
            GREEN = 0;
            VIB = 0;
        }
    } else if(podatak == pagerID) {
        if(message_header == '*' && STANJE == CEKANJE || STANJE ==
IZVANOPSEGA)
            STANJE = PROZVAN;

        if(message_header == '#' && STANJE == PROZVAN || STANJE ==
IZVANOPSEGA) {
            STANJE = CEKANJE;
            RED = 0;
            GREEN = 0;
            VIB = 0;
        }
        message_header = 0x00;
    }
}

void interrupt prekidna_rutina(){
    if (TMR0IE && TMR0IF){
        TMR0IF = 0;
        if(STANJE != CEKANJE && BATERIJA == NIJEPRAZNA){
            brojacLED++;
            if(brojacLED == 8){
                play++;
                brojacLED = 0;
                if(STANJE == PROZVAN){
                    GREEN = 1;
                    RED = 0;
                } else if (STANJE == IZVANOPSEGA){
                    RED = 1;
                    GREEN = 0;
                }
                led_show();
            }
        }
    } else if(TMR2IE && TMR2IF){
        TMR2IF = 0;
        if(STANJE == PROZVAN || STANJE == IZVANOPSEGA && BATERIJA ==
NIJEPRAZNA) {
            if(play < 2){
                sound++;
                VIB = 0;
                if(sound == 1) BUZZ = 1;
                if (sound == 2){
                    sound = 0;
                    BUZZ = 0;
                }
            } else if(play < 4)
                VIB = 1;
              else
```

```
                play = 0;
        }
    } else if (TMR1IE && TMR1IF){
        TMR1IF = 0;
        loop++;
        if (loop == 19){
           if (opsegID == 0x00)
                STANJE = IZVANOPSEGA;
            else
                opsegID = 0x00;
            loop = 0;
            GO = 1;
        }
    } else if (ADIE && ADIF){
        ADIF = 0;
        if (ADRESH > 55) {
            BATERIJA = PRAZNA;
            RED = 1;
            GREEN = 0;
            D1 = D2 = D3 = D4 = 0;
            prazna = 1;
        } else {
            BATERIJA = NIJEPRAZNA;
            if(prazna == 1){
                D1 = D2 = D3 = D4 = 1;
                RED = 0;
                GREEN = 0;
            }
            prazna = 0;
        }
    }
}

// Inicijalizacija stanja, usmjerenja pinova, pocetnog stanja pinova,
// AD konvertora, timera i prekida
void init() {
    // Pocetno stanje
    STANJE = CEKANJE;

    // Pocetno stanje baterije
    BATERIJA = NIJEPRAZNA;

    // Usmjerenja pinova
    TRISD = 0x00;           // PORTD je izlazni [RGB diode]
    TRISAbits.TRISA0 = 1;   // RA0 je ulazni [Napon baterije]
    TRISAbits.TRISA5 = 0;   // RA5 je izlazni [Vibrator]
    TRISCbits.TRISC3 = 0;   // RC2 je izlazni [Buzzer]
    TRISBbits.TRISB0 = 1;   // RB0 je ulazni [Data pin RF prijemnika]

    // Pocetna stanja pinova
    // Sve diode ugasene, nema boje
    D1 = 1;
    D2 = 1;
    D3 = 1;
    D4 = 1;
    RED = 0;
    GREEN = 0;
    BLUE = 0;

    VIB = 0;    // Vibrator iskljucen
    BUZZ = 0;   // Buzzer iskljucen
```

```
    // Inicijalizacija modula
    init_analog();
    init_buzz();
    init_batt();
    init_led();
    PEIE = 1;
    GIE = 1;
    if (DEBUG == 1)
        init_debug();
}

//  Inicijalizacija AD konvertora
void init_analog() {
    ADFM = 0;   // Lijevo poravnavanje rezultata

    PCFG3 = 1;  // RA0 analogni, RA1-7 digitalni
    PCFG2 = 1;  // Vref+ = Vdd, Vref- = Vss
    PCFG1 = 1;
    PCFG0 = 0;

    CHS2 = 0;   // Channel 0 (AN0)
    CHS1 = 0;
    CHS0 = 0;

    ADCS2 = 1;  // Interni RC oscilator
    ADCS1 = 1;
    ADCS0 = 1;

    ADON = 1;   // Ukljuivanje modula
    ADIE = 1;
}

// Inicijalizacija TIMER2 modula
void init_buzz() {
    T2CKPS1 = 1;    // Prescale 1:16
    T2CKPS0 = 1;

    PR2 = 125;
    TMR2ON = 1;     // Ukljucivanje Timer 2
    TMR2IE = 1;
}

// Inicijalizacija TIMER1 modula [Indikacija baterije, izlazka iz opsega]
void init_batt() {
    T1CKPS1 = 1; // Prescaler 1:8
    T1CKPS0 = 1;

    T1OSCEN = 0;
    TMR1CS = 0;

    TMR1ON = 1;
    TMR1IE = 1;
}

// Inicijalizacija Timer 0 modula
void init_led() {
    T0CS = 0;

    PSA = 0;    // Prescaler 1:8
    PS2 = 1;
```

```
    PS1 = 1;
    PS0 = 1;
    TMR0IE = 1;
}

// Inicijalizacija serijske komunikacije sa PC-om
void init_debug() {
    BRGH = 1;
    SPBRG = 51;
    SYNC = 0;
    SPEN = 1;
    TXEN = 1;
}

// LED rotacija
void led_show() {
    if (D1 == 0) {
        D1 = 1;
        D2 = 0;
    } else if(D2 == 0) {
        D2 = 1;
        D3 = 0;
    } else if(D3 == 0) {
        D3 = 1;
        D4 = 0;
    } else {
        D4 = 1;
        D1 = 0;
    }
}
```

Dodatak 2. Dnevnici rada

Mirzet Brkić

Datum	Vrijeme pocetka	Vrijeme zavrsetka	Trajanje [min]	Opis aktivnosti
09.12.2015.	22:45	23:20	35	Konsultacije sa kolegama
14.12.2015.	17:30	18:50	80	Detaljnije preciziranje teme, projektnih aktivnosti, komponenti
15.12.2015.	16:00	16:45	45	Test dometa nRF24L01+ modula
28.12.2015.	12:00	14:30	150	Logička shema – Centralni transmiter
29.12.2015.	00:00	02:00	120	PCB dizajn – Centralni transmiter
29.12.2015.	15:00	17:30	150	Logička shema – Pager
30.12.2015.	00:00	02:00	120	PCB dizajn – Pager
01.01.2016.	15:00	17:00	120	PCB dizajn – Pager
04.01.2016.	12:30	15:00	150	Test PCB dizajna, priprema komponenti
04.01.2016.	22:00	23:15	75	Izmjene u PCB dizajnu
01.02.2016.	17:00	20:00	180	Izrada PCB punjac – pokusaj 1
02.02.2016.	11:00	21:00	600	Izrada PCB pager, predajnik, punjac v2
03.02.2016.	10:00	18:00	480	Izrada PCB pager v2, punjac v3
18.02.2016.	10:00	17:00	420	Isprakve PCB
19.02.2016.	09:00	12:00	180	Ispravke PCB
21.02.2016.	14:00	20:00	360	Protokol komuniciranja, Pager code
22.02.2016.	12:00	16:00	240	Protokol komuniciranja, Predajnik
22.02.2016.	18:00	20:00	120	Protokol komuniciaranj, Pager – pokusaj interrupta
26.2.2016.	10:00	15:00	300	Komunikacija, interrupt
9.4.2016.	10:00	14:40	280	PC komunikacija
2.6.2016.	18:00	19:15	75	Korekcije koda za Pager
18.6.2016.	17:00	18:25	85	Sitne ispravke u kodu, eagle ispravke
27.6.2016.	15:00	19:00	240	Novi kod za Pager, komunikacija
28.6.2016.	15:00	17:00	120	Ispravke PC aplikacija
29.6.2016.	14:00	16:30	150	Ispravke u kodu: Pager, transmiter, PC

Almir Bešić

Datum	Vrijeme pocetka	Vrijeme zavrsetka	Trajanje [min]	Opis aktivnosti
09.12.2015.	22:45	23:20	35	Konsultacije sa kolegama
14.12.2015.	17:30	18:50	80	Detaljnije preciziranje teme, projektnih aktivnosti, komponenti
15.12.2015.	16:00	16:45	45	Test dometa nRF24L01+ modula
28.12.2015.	12:00	14:30	150	Logička shema – Centralni transmiter
28.12.2015.	20:00	21:15	75	Baterijsko napajanje, punjenje - istraživanje
29.12.2015.	15:00	17:30	150	Logička shema – Pager
04.01.2016.	12:30	15:00	150	Test PCB dizajna, priprema komponenti
01.02.2016.	17:00	20:00	180	Izrada PCB punjac – pokusaj 1
02.02.2016.	11:00	21:00	600	Izrada PCB pager, predajnik, punjac v2

03.02.2016.	10:00	18:00	480	Izrada PCB pager v2, punjac v3
18.02.2016.	10:00	17:00	420	Isprakve PCB
19.02.2016.	09:00	12:00	180	Ispravke PCB
21.02.2016.	14:00	20:00	360	Protokol komuniciranja, Pager code
22.02.2016.	12:00	16:00	240	Protokol komuniciranja, Predajnik
26.2.2016.	10:00	15:00	300	Komunikacija, interrupt
5.3.2016.	21:00	22:30	90	Komunikacija C# i serial port - istraživanje
6.3.2016.	19:00	21:30	150	Programiranje PC aplikacije
9.4.2016.	10:00	14:40	280	PC komunikacija
2.6.2016.	18:00	19:15	75	Korekcije koda za Pager
18.6.2016.	17:00	18:25	85	Sitne ispravke u kodu, eagle ispravke

Semir Berković

Datum	Vrijeme pocetka	Vrijeme zavrsetka	Trajanje	Opis aktivnosti
09.12.2015.	22:45	23:20	35	Konsultacije sa kolegama
14.12.2015.	17:30	18:50	80	Detaljnije preciziranje teme, projektnih aktivnosti, komponenti
15.12.2015.	16:00	16:45	45	Test dometa nRF24L01+ modula
28.12.2015.	12:00	14:30	150	Logička shema – Centralni transmiter
29.12.2015.	15:00	17:30	150	Logička shema – Pager
04.01.2016.	12:30	15:00	150	Test PCB dizajna, priprema komponenti
01.02.2016.	17:00	20:00	180	Izrada PCB punjac – pokusaj 1
02.02.2016.	11:00	21:00	600	Izrada PCB pager, predajnik, punjac v2
03.02.2016.	10:00	18:00	480	Izrada PCB pager v2, punjac v3
21.02.2016.	14:00	20:00	360	Protokol komuniciranja, Pager code
03.04.2016	18:00	20:00	120	Istrazivanje načina rada RFID modula
25.04.2016	11:00	12:20	80	Testiranje RFID modula
28.06.2016	15:00	18:00	180	Pokušaj uspostavljanja SPI komunikacije
29.06.2016	14:10	15:50	100	Pisanje dokumentacije
30.06.2016	10:00	11:10	70	Finalni radovi na projektu i dovršavanje dokumentacije