

Internet stvari

Ispitni zadatak

Izveštaj o projektu “Pametni dom”

Profesor:

prof. dr Marko Tanasković

Asistent:

Uroš Dragović

Student:

Božidar Mladenović 2018240254

Datum:

11.05.2022

Univerzitet Singidunum

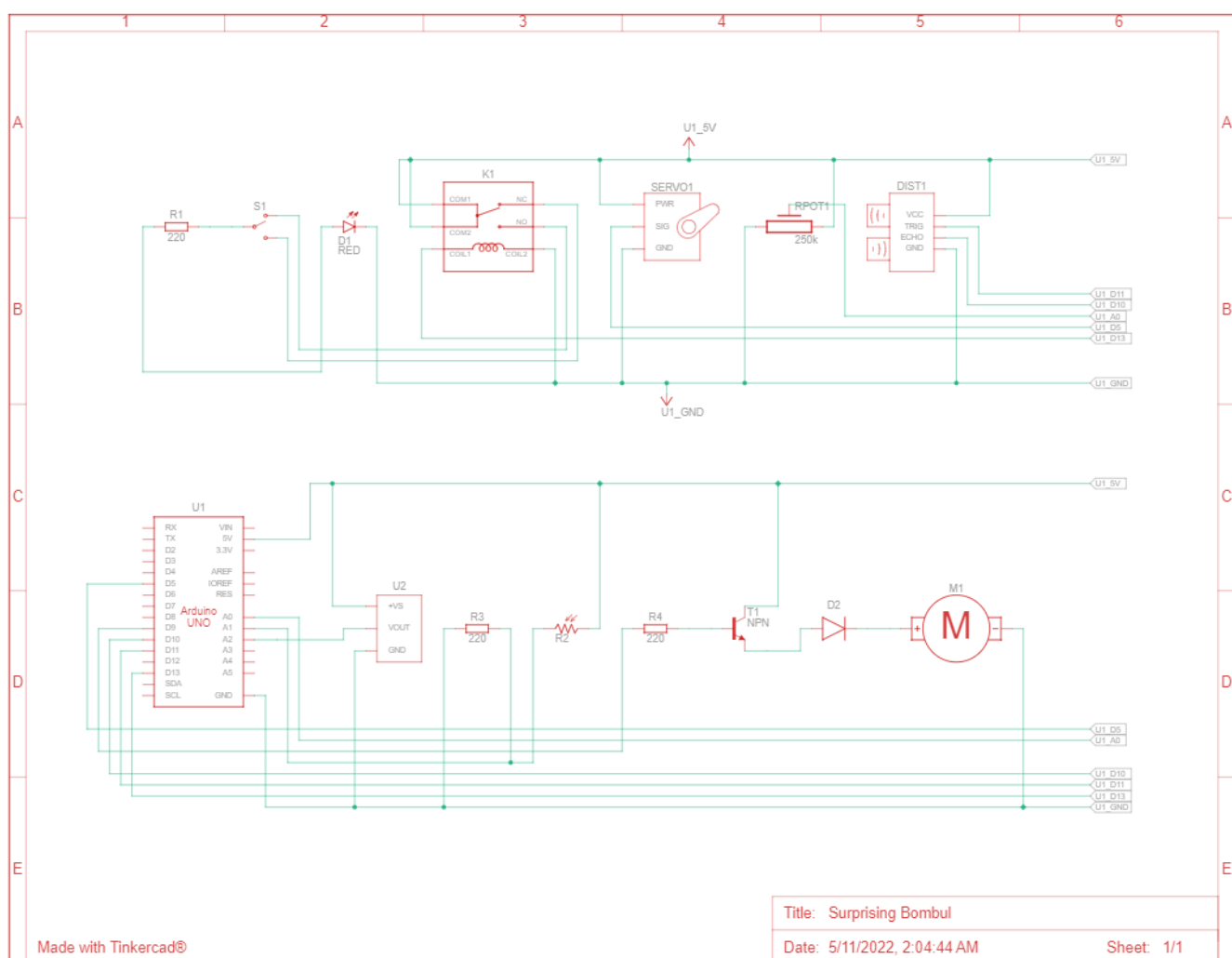
Beograd 2022

1	Uvod.....	3
2	Šematski i Slikovni prikaz	3
3	Implementacija	5
3.1	Prikaz koda za Arduino	5
3.2	Prikaz koda za web server	12
4	Literatura	16

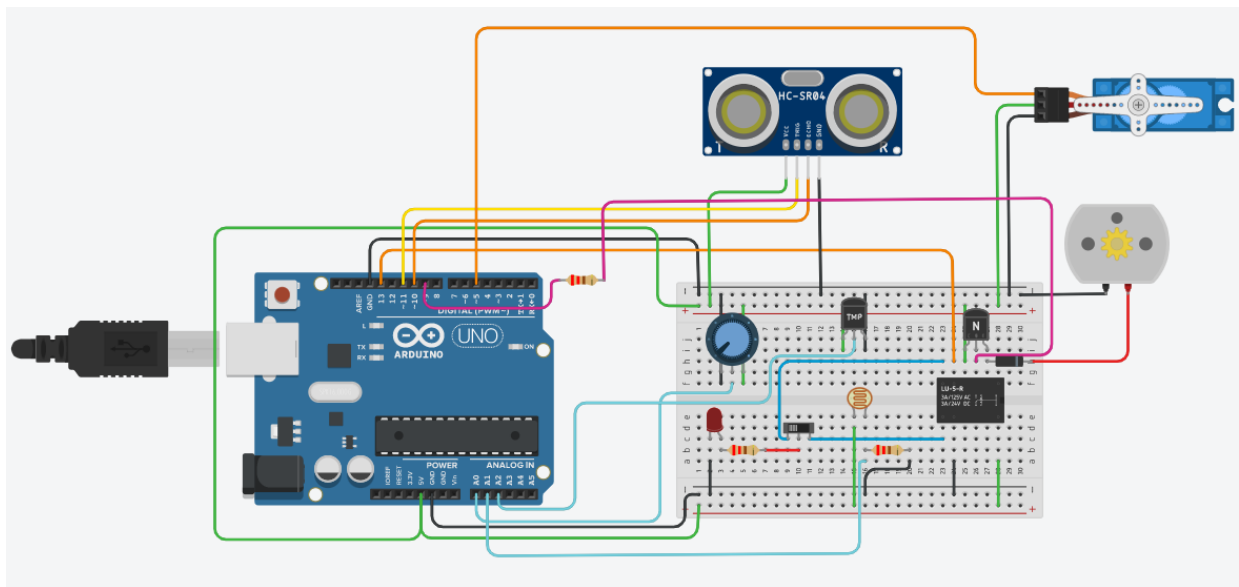
1 Uvod

U ovom izveštaju biće prikazana kompletna implementacija projekta „Pametni dom“ iz predmeta Internet stvari. Za implementaciju korišćeni programski jezici su: Python za implementaciju web servera (flask okvir) i modifikovana verzija programskog jezika C za konfigurisanje Arduina. Naravno pored računara korišćen je i Arduino mikrokontroler sa potrebnim komponentama i senzorima.

2 Šematski i Slikovni prikaz

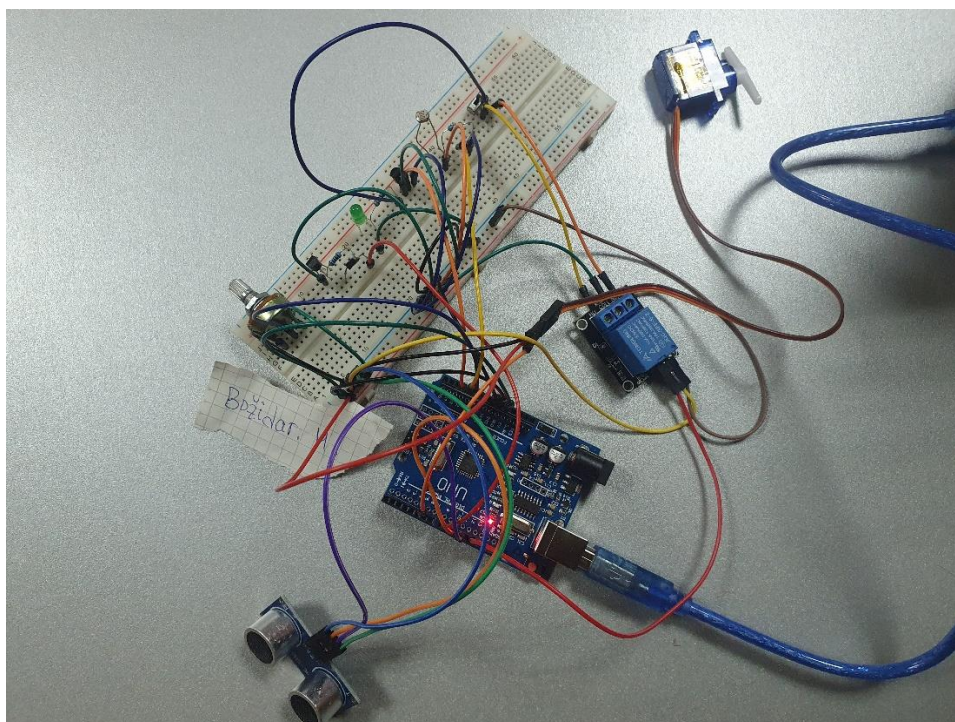


(Slika 1. Šematski prikaz zadatka)



(Slika 2. Slikovni prikaz zadatka)

Na slici 2 se može videti jedan Arduino Uno sa dodatim komponentama i senzorima i njihovim povezivanjem. Od komponenata u ovom projektu nalaze se: jedna dioda koja simulira sijalicu i koja može da se uključuje i isključuje uz pomoć fizičkog prekidača i uz pomoć releja preko web servera, jedan DC motor koji simulira rad ventilacije koja može da se kontroliše uz pomoć jednog potenciometra i takođe uz pomoć web servera i jedan servo motor koju služi da simulira jedna vrata koja mogu da se otvaraju i zatvaraju uz pomoć ultrasoničnog senzora koji računa razdaljinu i ponovo uz pomoć web servera. Pored ultrasoničnog senzora koji meri razdaljinu neke osobe ili objekta od senzora i po tome otvara i zatvara vrata, u ovom projektu postoje i senzor za temperaturu koju meri temperaturu u prostoriji kao i fotootpornik koji služi da pokaže koliko je neka prostorija osvetljena.



(Slika 3. Izgled fizičkog uređaja)

3 Implementacija

U ovom delu biće prikazani i detaljno objašnjeni delovi koda i za arduino (Modifikovani C) i za web server(Python).

3.1 Prikaz koda za Arduino

```
//Dodavanje TimerOne biblioteke u projekat
#include <TimerOne.h>

//Definisane Id-a Arduina
#define ARDUINO_ID 0

//Definisane konstante za pinove
#define RELAY_PIN 13
#define DC_MOTOR 9
#define T_ULTRASONIC_SENSOR 11
#define E_ULTRASONIC_SENSOR 10
#define SERVO_MOTOR 5

#define POTENT_PIN A0
#define BRIGHTNESS_PIN A1
#define TEMP_PIN A2

//Definisane konstante za stanja za vremenski prekid
#define TRANSMIT 0
#define RECEIVE 1
#define DOOR_STATE 2
#define WAIT 3

//Definisane i inicijalizovanje potrebnih promenljivih
int relay_state = LOW;
int potent_value = 0;

int old_potent_value = 0;

unsigned long duration;
float distance;
int distance_int = 0;
int door_serial = 0;
int is_opened = 0;
const unsigned long time_interrupt = 2000;
int time_counter = 0;
int current_state;

int relay_state_counter = 0;
int open_door_counter = 0;
```

(Slika 4. Prikaz definisanja potrebnih konstanti i promenljivih)

Na slici 4 se mogu videti definisane konstante za id arduina, konstante za digitalne i analogne pinove mikrokontrolera kao i konstante za jednu mašinu stanja koja se koristi za vremenski prekid tačnije ultrasonični senzor. Pored konstanti na slici se vide i potrebne promenljive koje su potrebne za rad aplikacije i to su: promenljiva za stanje diode, promenljiva za vrednost potencijometra, promenljiva za staru

vrednost potencijometra koja služi da spreči preklapanje komandi za dc motor sa web servera i sa samog potencijometra , promenljive koje su potrebne za ultrasonični senzor i vremenski prekid i na kraju promenljive u kojima treba da se skladišti podatak koliko puta su se vrata otvorila i koliko puta se promenilo stanje na releju.

```
void setup()
{
    //Pokretanje serijske komunikacije
    Serial.begin(9600);

    //Postavljanje modova za ulaz i izlaz na potrebnim pinovima
    pinMode(RELAY_PIN, OUTPUT);
    pinMode(DC_MOTOR, OUTPUT);
    pinMode(T_ULTRASONIC_SENSOR, OUTPUT);
    pinMode(E_ULTRASONIC_SENSOR, INPUT);
    pinMode(SERVO_MOTOR, OUTPUT);

    //Setovanje diode na pocetno stanje
    digitalWrite(RELAY_PIN, relay_state);

    //Inicijalizovanje tajmera na 0.002 sekunde
    Timer1.initialize(time_interrupt);
    //Tajmeru se povezuje prekid koji ce da pozove funkciju za vremenski prekid
    Timer1.attachInterrupt(timer_function);

    current_state = TRANSMIT;
}
```

(Slika 5. Prikaz setup funkcije)

Funkcija Setup je ugradjena funkcija koja se prva pokreće i koja služi za inicijalizaciju celog projekta, na slici 5 se mogu videti komande kao sto su "pinMode" koje postavljaju modove za pinove podrazumevano u arduinu svi digitalni pinovi imaju INPUT mod koji služi za unos, pored INPUT moda u ovom projektu je korišćen i OUTPUT mod koji služi za čitanje. "Serial.begin" komanda je tu da bi se omogućila serijska komuikacija sa računarom, zatim „timer1.initialize()“ koja služi za inicijalizaciju tajmera na 0,002 sekunde i komanda „Timer1.attachInterrupt()“ koja služi da tajmeru poveže prekid koji će da poziva funkciju prekida. Na kraju imamo current_state promenljivu koja služi da postavimo početno stanje u vremenskom prekidu.

```

void loop()
{
    //Uslov koji proverava da li postoji nesto na serijskoj liniji
    if (Serial.available() > 0) {
        //U promenljivu command smestamo string sa serijske linije koji se zavrшава simbolom ;
        String command = Serial.readStringUntil(';');
        //U promenljivu i smestamo indeks prve dvotacke u stringu
        int i = command.indexOf(':');

        if (i > 0) {
            //U promenljivu arduinoId smestamo id arduina koji se nalazi na pocetku stringa
            String arduinoId = command.substring(0, i);
            //String id prebacujemo u integer
            int id = arduinoId.toInt();

            //U promenljivu write_read smestamo treci karakter stringa koji služi da kaže
            //da li se radi o citanju ili pisanju
            char write_read = command.charAt(i + 1);

            if (write_read == 'W') {
                // Zahtev za upisivanje

                int s = i + 3;
                //Pronadji indeks dvotacke nakon indeksa 4
                i = command.indexOf(':', s);
                //Dohvati sve do te dvotacke
                String pin = command.substring(s, i);
                //Dohvati string na indeksu 6
                String value = command.substring(i + 1);

                if (id != ARDUINO_ID) {
                    // posalji dalje

```

(Slika 6. Prvi deo loop funkcije)

Funkcija loop je takođe ugrađena funkcija koja se pokreće odmah nakon setup funkcije i na slici 6 je prikazan prvi deo te funkcije. U ovoj funkciji prvo se nalazi uslov koji ispituje da li postoji nešto poslato arduinu preko serijske linije i ako ima onda se u nastavku nalazi kod koji poruku koja je određenog formata secka u podatke koji su potrebni. Pa se tako na kraju dobijaju podaci o tome koji je id arduina, koji je pin i koja vrednost je poslata.

```

    } else {
        //Ako je kroz string poslat RELAY_PIN putem serijske komunikacije
        //zameniti stanje diode na tom pinu
        if (pin == "RELAY_PIN") {
            //Menjanje stanja promenlive za stanje diode
            relay_state = !relay_state;
            //Inkrementiranje promenljive koja služi za podatak koliko je puta relej promenio stanje
            relay_state_counter++;
            //Upis novog stanja na pin diode
            digitalWrite(RELAY_PIN, relay_state);
            //Ako je kroz string poslat DC_MOTOR pin putem serijske komunikacije
            //promeniti brzinu motora(ventilatora)tom pinu
        } else if (pin == "DC_MOTOR") {
            //Prebacivanje vrednosti motora iz string u integer tip
            int value_int = value.toInt();
            //Mapiranje vrednosti u odgovarajuci opseg
            int m_speed = map(value_int, 0, 100, 0, 255);
            //Upis nove brzine motora
            analogWrite(DC_MOTOR, m_speed);
            //Ako je kroz string poslat SERVO_MOTOR pin putem serijske komunikacije
            //postaviti promenljivu door_serial na 1
        } else if (pin == "SERVO_MOTOR") {
            door_serial = 1;
        }
    }
}
}
//Pozivanje funkcije za promenu brzine dc motora preko potencijometra
speed_of_dc_motor();
}

```

(Slika 7. Drugi deo loop funkcije)

Na slici 7 je prikazan drugi deo loop funkcije i u njemu se radi provera koji je pin poslat i postavljanje novih vrednosti, pa tako na primer ako je poslat pin „RELAY_PIN“ onda će se izvršiti kod koji menja stanje diode, ako je poslat pin „DC_MOTOR“ uz njega se šalje i vrednost od 0 do 100 i onda se izvršava kod koji pali motor koji se vrti brzinom koja je definisana tom poslatom vrednošću i ako je poslat pin „SERVO_MOTOR“ onda se promenljiva „door_serial“ setuje na 1 što znači da je poslat zahtev da se promeni stanje vrata preko serijske komunikacije. Na kraju loop funkcije se poziva funkcija koja kontroliše dc motor preko potencimetra.

```
//Definisanje funkcije za promenu brzine dc motora preko potencimetra
void speed_of_dc_motor()
{
    //Smestanje vrednost potencimetra u promenljivu
    potent_value = analogRead(POTENT_PIN);

    //Definisanje delta promenljive koja služi za toleranciju potencimetra
    int delta = 10;
    //Mapiranje vrednosti u odgovarajuće opsege
    int motor_speed = map(potent_value, 0, 1023, 0, 255);

    //Uslov koji resava preklapanja promene brzine motora preko potencimetra i
    //preko serijske komunikacije
    if (!(motor_speed >= (old_potent_value - delta) && motor_speed <= (old_potent_value + delta))) {
        analogWrite(DC_MOTOR, motor_speed);
        old_potent_value = motor_speed;
    }
}
```

(Slika 8. Funkcija koja kontroliše dc motor preko potencimetra)

Na slici 8 je prikazana funkcija koja kontroliše vrednost dc motora tako što čita vrednost potencimetra i na pin motora upisuje vrednost, međutim pošto motor može da se kontroliše i serijskom komunikacijom ovde postoji još jedan promenljiva kojom se sprečava mešanje komandi tako što se upoređuju stara i nova vrednost i promenljiva delta koja je tu zbog nekih odstupanja potencimetra.

```
//Funkcija koja služi za vremenski prekid
void timer_function() {
    time_counter++;
    switch (current_state) {
        //Stanje u kome senzor za merenje razdaljine transmituje signal
        case TRANSMIT:
            //Generisanje trigger signala
            digitalWrite(T_ULTRASONIC_SENSOR, HIGH);
            //Posle 8 milisekundi promeniti stanje u RECEIVE
            if (time_counter % 4 == 0) {
                current_state = RECEIVE;
            }
            break;
        //Stanje u kome senzor za merenje razdaljine prima signal koji transmitovao
        case RECEIVE:
            //Trigger signal se gasi
            digitalWrite(T_ULTRASONIC_SENSOR, LOW);
            //Osluskuje se echo port i u promenljivu se upisuje vreme koliko je signal putovao
            duration = pulseIn(E_ULTRASONIC_SENSOR, HIGH);
            //Racuna se distanca objekta ili osobe od senzora
            distance = duration * 0.034 / 2;
            current_state = DOOR_STATE;
            break;
    }
}
```

(Slika 9. Funkcija prekida prvi deo)

Na slici 9 prikazana je funkcija prekida koja se pokreće na svakih 2000 mikrosekundi ili 0,002 sekunde i ona se koristi za kontrolisanje ultrasoničnog senzora koji daje podatke o razdaljini nekog predmeta ili osobe od senzora i koji služi za otvaranje vrata ako se neka osoba približi i takođe u ovoj funkciji se nalazi kod koji na svaki minut šalje podatke web serveru. Kao što se vidi na slici 9 u ovoj funkciji se nalazi jedna mašina stanja koja ima 4 stanja, u prvom stanju ultrasonični senzor transmituje signal 8 milisekundi i nakon toga se prebacuje u drugo stanje gde se taj signal čita nazad i onda meri koliko je vremena bilo potrebnu signalu da se odbije od nečega i da se vrati, zatim se od dobijenog vremena uz pomoć formule računa koliko je neko ili nešto udaljeno od senzora.

```
//Stanje koje služi za promenu stanja servo motora tj otvaranje i zatvaranje vrata
case DOOR_STATE:
    distance_int = int(distance);
    //Ako osoba pridje senzoru blize od 5 cm ili sa serijske linije dobijemo naredbu
    //za promenu stanja vrata i ako su vrata zatvorena, otvoriti vrata
    if (((distance_int <= 5) || (door_serial == 1)) && (is_opened == 0)) {
        //Inkrementiranje promenljive koja služi za informaciju koli puta su vrata otvorena
        open_door_counter++;
        analogWrite(SERVO_MOTOR, 250);
        //Setuje se promenljiva koja pokazuje da li su vrata otvorena na 1
        is_opened = 1;
        //Promenljiva koja služi da nam kaze da li imamo naredbu sa serijskog porta se vraća u pocetno stanje
        door_serial = 0;
    }
    //Ako se osoba udalji od senzora vise od 5 cm ili sa serijske linije dobijemo naredbu
    //za promenu stanja vrata i ako su vrata otvorena, zatvoriti vrata
    else if (((distance_int > 5) || (door_serial == 1)) && (is_opened == 1)) {;
        analogWrite(SERVO_MOTOR, 0);
        //Setuje se promenljiva koja pokazuje da li su vrata otvorena na 0
        is_opened = 0;
        door_serial = 0;
    }

    current_state = WAIT;
    break;
//Stanje u kome timer čeka da ponovo transmituje signal
case WAIT:
    //Nakon 1 sekunde ponovo prebaci u stanje za TRANSMIT
    if (time_counter % 500 == 0) {
        current_state = TRANSMIT;
    }
    break;
}
```

(Slika 10. Funkcija prekida drugi deo)

Na slici 10 prikazana su ostala dva stanja mašine stanja koja se nalazi u funkciji vremenskog prekida, pa tako kada se odredi distanca između nečega i senzora, prelazi se u treće stanje u kome se nalazi kod za kontrolisanje servo motora to jest otvaranja ili zatvaranje vrata i stanje 4 to je stanje u kome tajmer čeka da ponovo transmituje signal to jest čeka 1 sekundu.

```

//Uslov koji se izvršava svakog minuta i šalje podatke web serveru
if (time_counter % 15000 == 0) {
    //Dohvatanje vrednosti osvetljenja uz pomoc funkcije
    int bri_value = get_brightness_in_precentage();
    //Slanje vrednosti osvetljena web serveru
    send_data("BRIGHTNESS_PIN", (float)bri_value);

    //Dohvatanje vrednosti temperature uz pomoc funkcije
    float temp_value = get_temperature_in_c();
    //Slanje vrednosti temperature web serveru
    send_data("TEMP_PIN", temp_value);

    //Slanje ostalih vrednosti
    send_data("DOOR_COUNTER", open_door_counter);
    send_data("RELAY_COUNTER", relay_state_counter);
    send_data("LIGHT", relay_state);
    send_data("SERVO_MOTOR", is_opened);
}
}

```

(Slika 11. Funkcija prekida treći deo)

U ovom delu prekida funkcije nalazi se kod koji svaki minut šalje podatke web serveru, podaci koji se šalju su podaci o temperaturi, osvetljenju prostorije, koliko su se puta otvorila vrata, koliko puta se promenilo stanje na releju, da li su vrata otvorena itd. Svi ovi podaci se serveru šalju u određenom formatu kako bi ih na web serveru lako čitali. Podatke prebacujemo u odgovarajući format uz pomoć funkcije „send_data“ koja je prikazana na slici 12.

```

//Funkcija za kreiranje formata za slanje podataka serveru
void send_data(String pin, float value) {
    String s = String(ARDUINO_ID) + ":" + String(pin) + "|" + String(value) + ";";
    Serial.print(s);
}

```

(Slika 12. Funkcija koja podatke smešta u format za slanje)

```

//Funkcija za dohvatanje temperature
float get_temperature_in_c() {
    //Citamo vrednost sa senzora za temperaturu
    int analog_temp_value = analogRead(TEMP_PIN);

    //Formula za dobijanje temperature u C
    float real_temp = 500 * analog_temp_value / 1023.0f;

    return real_temp;
}

```

(Slika 13. Funkcija koja dohvata temperaturu sa senzora)

```

int get_brightness_in_precentage() {

    //Citanje vrednosti za fotootpornika
    int analog_bri_value = analogRead(BRIGHTNESS_PIN);
    //Mapiranje opsega
    int brightness_in_precentage = map(analog_bri_value, 1, 310, 0, 100);

    return analog_bri_value;

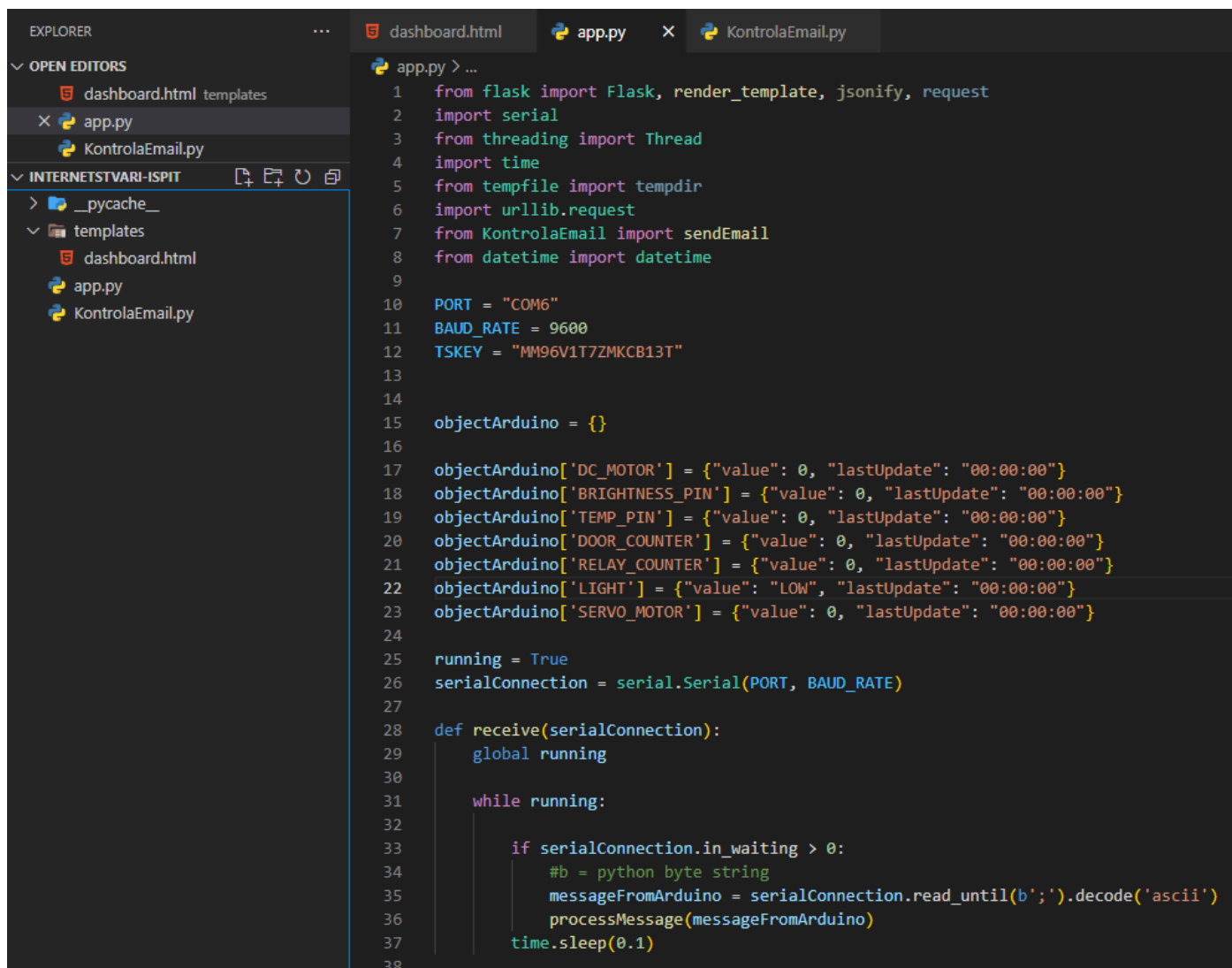
}

```

(Slika 14. Funkcija koja dohvata vrednost osvetljenja sa senzora)

Na slici 13 i 14 su prikazane funkcije koje se koriste za dohvatanje temperature sa senzora temperature i vrednosti osvetljenja sa fotootpornika.

3.2 Prikaz koda za web server



```
1 from flask import Flask, render_template, jsonify, request
2 import serial
3 from threading import Thread
4 import time
5 from tempfile import tempdir
6 import urllib.request
7 from KontrolaEmail import sendEmail
8 from datetime import datetime
9
10 PORT = "COM6"
11 BAUD_RATE = 9600
12 TSKEY = "MM96V1T7ZMKCB13T"
13
14
15 objectArduino = {}
16
17 objectArduino['DC_MOTOR'] = {"value": 0, "lastUpdate": "00:00:00"}
18 objectArduino['BRIGHTNESS_PIN'] = {"value": 0, "lastUpdate": "00:00:00"}
19 objectArduino['TEMP_PIN'] = {"value": 0, "lastUpdate": "00:00:00"}
20 objectArduino['DOOR_COUNTER'] = {"value": 0, "lastUpdate": "00:00:00"}
21 objectArduino['RELAY_COUNTER'] = {"value": 0, "lastUpdate": "00:00:00"}
22 objectArduino['LIGHT'] = {"value": "LOW", "lastUpdate": "00:00:00"}
23 objectArduino['SERVO_MOTOR'] = {"value": 0, "lastUpdate": "00:00:00"}
24
25 running = True
26 serialConnection = serial.Serial(PORT, BAUD_RATE)
27
28 def receive(serialConnection):
29     global running
30
31     while running:
32
33         if serialConnection.in_waiting > 0:
34             #b = python byte string
35             messageFromArduino = serialConnection.read_until(b';').decode('ascii')
36             processMessage(messageFromArduino)
37             time.sleep(0.1)
38
```

(Slika 15. Početak app.py fajla)

Web server je kreiran uz pomoć „flask“ okvira u python-u i sastoji se iz 3 fajla, glavni fajl app.py, fajl koji služi sa slanje email izveštaja i html fajla za dashboard. Na samo početku app.py fajla nalaze se razni importi koji su potrebni za ispravno funkcionisanje ovog web servera, zatim 3 konstante, to su PORT preko kog komuniciramo sa arduinom, BAUD_RATE i TSKEY uz pomoć kog komuniciramo sa thingspeak platformom pošto se sa ovog web servera podaci šalju tamo . Posle toga postoji jedan templejt objekta koji se ažurira podacima sa arduina, zatim funkcija „receive“ u kojoj uzimamo poruku sa serijal linije i prosleđujemo funkciji „processMessage“ koja služi za obradu podataka koju su poslali, ažuriranje objekta i slanje podataka na thingspeak.

```

def processMessage(message):
    # ODGOVOR : "ARDUINO_ID:PIN|VREDNOST;ARDUINO_ID:PIN|VREDNOST;"
    splitObjects = message[:-1].split(";")

    for obj in splitObjects:
        explode = obj[:-1].split(":")
        arduinoId = int(explode[0])
        pin = str(explode[1].split("|")[0])
        value = float(explode[1].split("|")[1])

        objectArduino[pin]['value'] = value

    urllib.request.urlopen('https://api.thingspeak.com/update?api_key={}&field1={}&field2={}&field3={}&field4={}'.format(TSKEY, objectArduino['TEMP_PIN']['value'], objectArduino['B

threadReceiver = Thread(target=receive, args=(serialConnection,))
threadReceiver.start()

threadReceiver = Thread(target=sendEmail)
threadReceiver.start()

```

(Slika 16. Funkcija „processMessage“)

Na slici 16 prikazana je funkcija processMessage koja služi da primljenu poruku pripremi i ažurira u objekat i nakon toga da podatke iz te poruke pošalje na thingspek platformu. Na web serveru imamo dve niti na jednoj se pokreće glavna funkcija receive koja se stalno vrti i svaki minut funkciji procesMessage šalje poruku sa arduino, druga nit je email izveštaj i ona se pokreće jednom u 24 sata.

```

59 app = Flask(__name__)
60
61 @app.route('/')
62 def dashboard():
63     global objectArduino
64     return render_template("dashboard.html", data=objectArduino)
65
66 @app.route('/change/<pin_id>', methods=['GET'])
67 def change(pin_id):
68     global serialConnection
69     global objectArduino
70     text = getWriteMessageWithoutValue(0, pin_id)
71     if (pin_id == 'RELAY_PIN'):
72         objectArduino['RELAY_COUNTER']['lastUpdate'] = str(datetime.now())
73     else:
74         objectArduino[pin_id]['lastUpdate'] = str(datetime.now())
75     serialConnection.write(text.encode('ascii'))
76     return render_template("dashboard.html", data=objectArduino)
77
78 @app.route('/setMotorSpeed/<pin_id>/<value>', methods=['GET'])
79 def setMotorSpeed(pin_id, value):
80     global serialConnection
81     global objectArduino
82     text = getWriteMessage(0, pin_id, value)
83     objectArduino[pin_id]['lastUpdate'] = str(datetime.now())
84     serialConnection.write(text.encode('ascii'))
85     return render_template("dashboard.html", data=objectArduino)
86
87 def getWriteMessage(controllerId, pin, value):
88     return str(controllerId) + ":W:" + str(pin) + ":" + str(value) + ";"
89
90 def getWriteMessageWithoutValue(controllerId, pin):
91     return str(controllerId) + ":W:" + str(pin) + ";"
92
93 if __name__ == "__main__":
94     app.run(port=5000, debug=True)

```

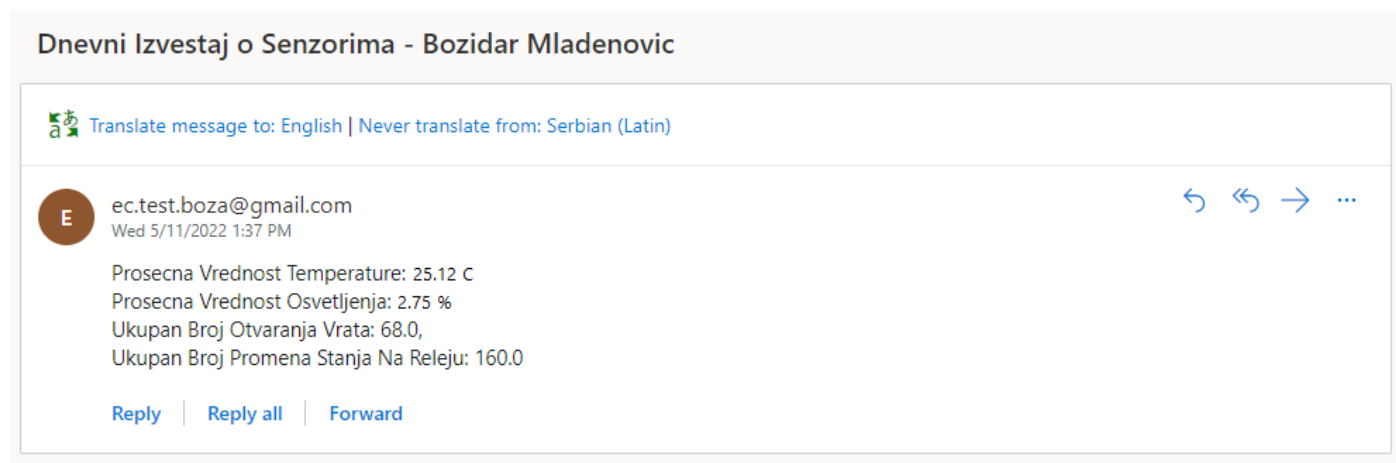
(Slika 17. Rute aplikacije)

Web server aplikacije ima 3 rute i to su početna ruta koja vodi na dashboard.html gde se šalje i objekat ažuriran podacima, zatim get rutu „change“ koja služi za promenu stanja releja i servo motora tako što se odlaskom na tu rutu pokreće funkcija koja na serijskoj liniji arduino šalje komandu i get rutu „setMotorSpeed“ uz pomoć koje se kontroliše dc motor na arduino isto preko serijske linije.

```
7 def sendEmail():
8     while True:
9         CHANNEL_ID = 1727854
10        TS_KEY = "IIQHkU7U0V65VD2A"
11
12        resultFromTs = requests.get("https://api.thingspeak.com/channels/{}/feeds.json?api_key={}".format(CHANNEL_ID, TS_KEY))
13
14        data = resultFromTs.json()['feeds']
15
16        sumTemp = 0
17        sumBri = 0
18        totalDoor = 0
19        totalRelay = 0
20
21        for one in data:
22            sumTemp += float(one['field1'])
23            averageTemp = sumTemp / len(data)
24
25            sumBri += float(one['field2'])
26            averageBri = sumBri / len(data)
27
28            totalDoor += float(one['field3'])
29
30            totalRelay += float(one['field4'])
31
32        server = smtplib.SMTP('smtp.gmail.com', 587)
33        server.starttls()
34
35        EMAIL_ADDRESS = input("Unesite email adresu posiljioca: ")
36        PASSWORD = input("Unesite sifru posiljioca: ")
37
38        DESTINATION_EMAIL_ADDRESS = input("Unesite email adresu primaoca: ")
39
40        resCode = server.login(EMAIL_ADDRESS, PASSWORD)
41
42        subject = "Dnevni Izvestaj o Senzorima - Bozidar Mladenovic"
43
44        body = "Prosečna Vrednost Temperature: {} C, \n".format(round(averageTemp, 2))
45        body += "Prosečna Vrednost Osvetljenja: {} %, \n".format(round(averageBri, 2))
46        body += "Ukupan Broj Otvaranja Vrata: {}, \n".format(totalDoor)
47        body += "Ukupan Broj Promena Stanja Na Releju: {} \n".format(totalRelay)
48
49        fullEmail = "Subject: {}\n\n{}".format(subject, body)
50
51        resCode = server.sendmail(from_addr=EMAIL_ADDRESS, to_addrs=DESTINATION_EMAIL_ADDRESS, msg=fullEmail)
52
53        server.quit()
54
55        time.sleep(86400)
```

(Slika 18. Funkcija za slanje email izveštaja)

Na slici 18 prikazan je funkcija za slanje email izveštaja tako što se preko thingspeak api-a dohvataju podaci zatim se računaju ukupne i srednje vrednosti i onda se te vrednosti šalju kao email poruka na svaka 24 sata.



(Slika 19. Izgled jednog email izveštaja)

```
templates > dashboard.html > html > body > div.container > div > table.table > tbody > tr > td.align-middle
1 <html>
2 <head>
3   <meta charset="utf-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1">
5   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTlfspd3yD65Vohhpuu"
6   <title>IoT Smart House</title>
7
8 </head>
9 <body>
10   <h1>Dashboard</h1>
11   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-Mrcw6ZMFY1zcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" cr
12   <div class="container">
13     <div>
14       <table class="table">
15         <thead class="thead-light">
16           <tr>
17             <th class="align-middle" scope="col">Temperatura</th>
18             <th class="align-middle" scope="col">Osvetljenost</th>
19             <th class="align-middle" scope="col">Broj Otvaranja Vrata</th>
20             <th class="align-middle" scope="col">Broj Promena Stanja Svetla</th>
21           </tr>
22         </thead>
23         <tbody>
24           <tr>
25             <td class="align-middle">
26               {{ data['TEMP_PIN']['value'] }} C
27             </td>
28             <td class="align-middle">
29               {{ data['BRIGHTNESS_PIN']['value'] }} %
30             </td>
31             <td class="align-middle">
32               {{ data['DOOR_COUNTER']['value'] }}
33             </td>
34             <td class="align-middle">
35               {{ data['RELAY_COUNTER']['value'] }}
36             </td>
37           </tr>
38         </tbody>
39       </table>
40
41       <br>
42       <br>
43
44       <table class="table">
45         <thead class="thead-light">
46           <tr>
47             <th class="align-middle" scope="col">Komponenta</th>
48             <th class="align-middle" scope="col">Opcija</th>
49             <th class="align-middle" scope="col">Vrednost</th>
```

(Slika 20. Izgled „dashboard.html“ fajla)

Dashboard

Temperatura	Osvetljenost	Broj Otvaranja Vrata	Broj Promena Stanja Svetla
20.5 C	0.0 %	3.0	6.0

Komponenta	Opcija	Vrednost	Azurirano
Svetlo	Promeni Stanje	Iskljuceno	2022-05-11 13:53:37.154391
Ventilator	Promeni Brzinu	<input type="text" value="0"/>	2022-05-11 13:53:52.073072
Vrata	Promeni Stanje	Zatvorena	2022-05-11 13:54:22.697216

(Slika 21. Izgled „Dashboard“ stranice)

Channel ID: 1727854
Author: mwa0000026464764
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

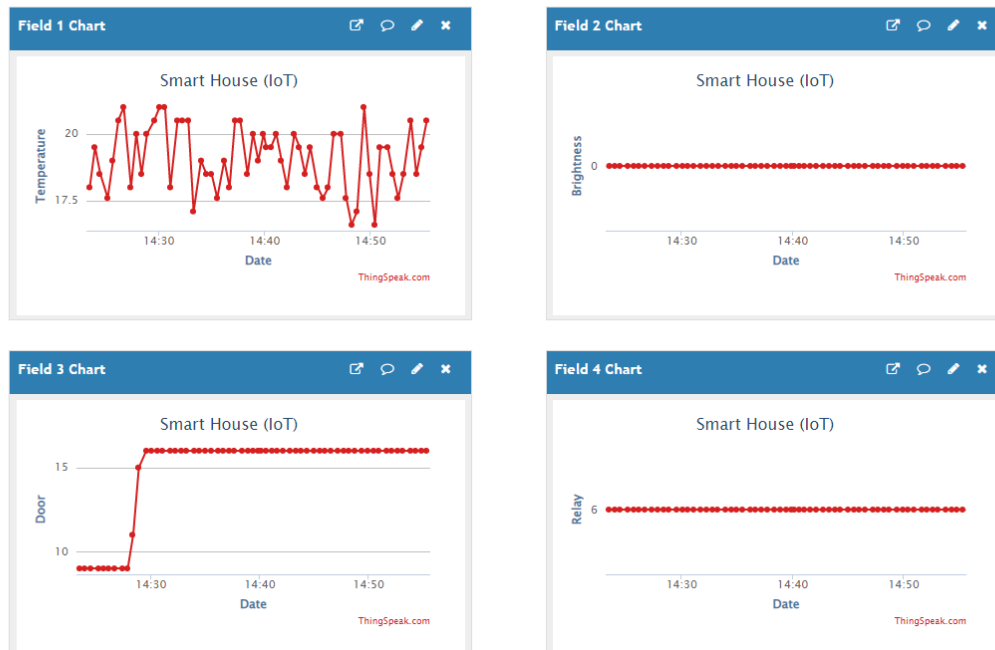
+ Add Visualizations + Add Widgets Export recent data

MATLAB Analysis

MATLAB Visualization

Channel Stats

Created: 2 days ago
Last entry: about 7 hours ago
Entries: 195



(Slika 22. Izgled „ThingSpeak“ platforme)

4 Literatura

- Udžbenik „Internet Stvari“ , Marko Tanasković, Beograd 2020.
- Materijali sa vežbi i predavanja
- <https://www.tinkercad.com/dashboard>