Univerzitet Singidunum
Tehnički fakultet

Arduino IoT sistem “Pametne kuće”

- Projektni rad -

Predmet: **Internet stvari**

Profesor:
prof. dr Marko Tanasković

Asistent:
Uroš Dragović

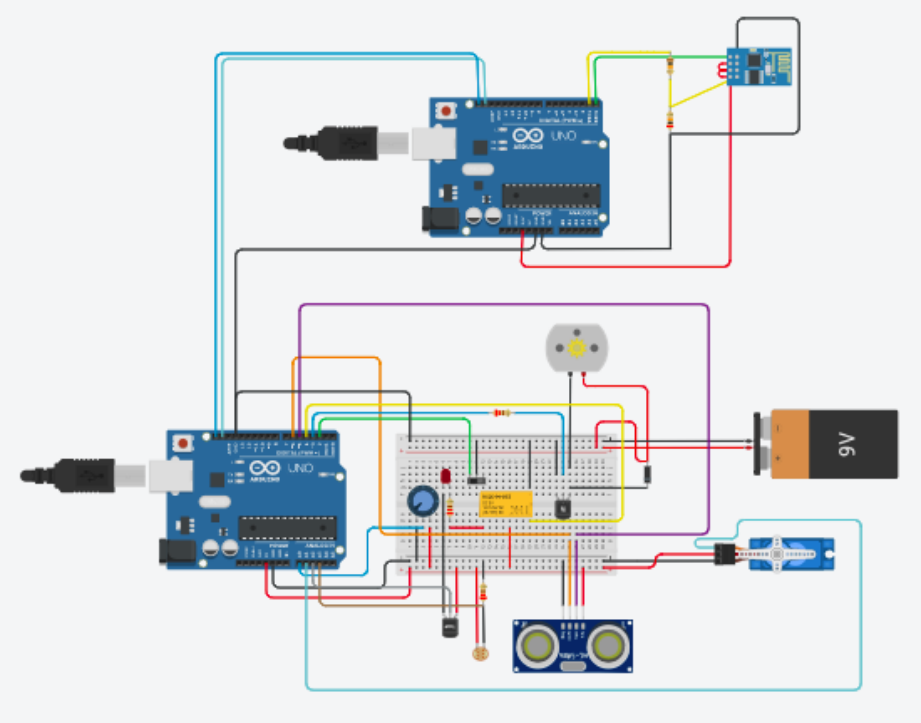
Student:
Marko Dojkić 2018/201682

Beograd, 2022. godine

Sadržaj

1. Uvod, šema povezivanja sistema

Ovaj sistem predstavlja IoT sistem za upravljanje "pametnom kućom". Svrha ovog sistema je mogućnost daljinskog upravljanja relejem (na koji je povezana dioda, a u realnosti bi bilo neko svetlo), brojem obrtaja DC motora (koji simulira ventilaciju) i rotorom servo motora (simulira otvaranje/zatvaranje vrata). Takođe sistem omogućava očitavanje osvetljenja (fotooptornik) i temperature (pomoću senzora TMP36). Ceo projekat je realizovan pomoću „Tinkercad" simulatora i dostupan za testiranje na sledećem linku: https://www.tinkercad.com/things/eDWHRQvkpOo?sharecode=P2-BrF4wLYwoFGrKoiOZ3-AAoDhFk-rC3U4bJFm-3jQ. Web server je programiran u Python programskom jeziku pomoću Flask okvira za razvoj web aplikacija hostovan na Heroku platformi i dostupan na sledećem linku: https://iot-ispit-python-webserver.herokuapp.com Takođe ovaj web server ima mogućnost slanje dnevnog izveštaja očitavanja senzora, koji se preko istog šalju na Thinkspeak platformu. Izveštaj sadrži broj otvaranja vrata, promena stanja na releju, prosečnu temperaturu i osvetljenost.

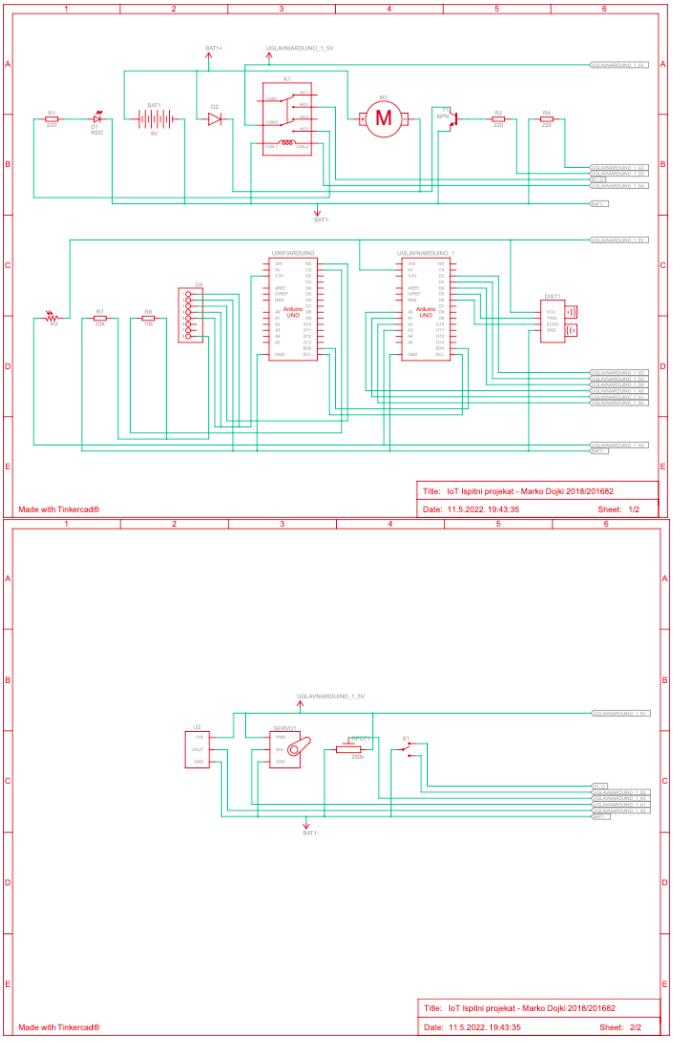


Slika 1:Prikaz ugašenog sistema na Tinkercad platformi

Za potrebe projekta korišćena su dva **Arduino Uno R3** uređaja na koji su povezane sledeće komponente:

- Na "WifiArduino" koji služi za komunikaciju sa Web serverom je povezan WiFi modul (ESP8266), a SDA i SDL pinovi su vezani na drugi da bi se ostvarila I2C veza. WiFi modul je obezbeđen sa 2 otpornika čija je otpornost 20 kΩ i 10 kΩ. Drugi "GlavniArduino_1" sadrži sledeće komponente:
    1) DPDT relej koji pali, odnosto gasi crvenu diodu. Dioda je obezbeđena otpornikom od 220Ω
    2) Klizni prekidač omogućava direktno upravljanje relejem, dok se istom digitalno upravlja pomoću komande sa servera ("relaySwitch")

3) DC motor koji simulira ventilaciju. Brzina DC motora se direktno zadaje analognim upisom preko NPN tranzistora (koji ograničava struju). On se napaja baterijom od 9V
4) Potencijometar otpornosti 250 kΩ čijim se obrtanjem menja i brzina motora. Vrednost koja se dobija sa istog je u opsegu od 0 do 1023, a ta vrednost se mapira na vrednost za NPN tranzisor (opsed od 0 do 255). Digitalni način upravljanja direktno šalje ovu vrednost komandom "$changeVentilationSpeed$<željena_brzina_u_navedenom_opsegu>"
5) Ultrazvučni senzor daljine, koji kada detektuje da se neki objekat nalazi na daljini manjoj od 5 cm šalje aktivira funkciju za otvaranje vrata, a kada se isti odalji aktivira funkciju za zatvaranje vrata. Digitalno je takođe moguće promeniti stanja vrata (otvoriti zatvorena i obrnuto) pomoću komande "openCloseDoors" dobijene sa servera
6) Fotootpornik koji je obezbeđen otpornikom od 220 Ω, a služi kao senzor osvetljenja u prostoriji (vrednost je pretvorena u lx korišćenjem formule) Vrednost se takođe salje i u % (u odnosu na ustanovljeni opseg od 2 do 988 koji zavisi od otpornosti ovog fotootpornika)
7) Senzor temperature TMP36 (vrednost je pretvorena u °C, a ona očitava temperaturu u opsegu od -50 °C do 50 °C)



Slike 2 i 3:Šema povezivanja sistema

2. Prikaz i objašnjenje koda

Kod Arduina koji komunicira sa serverom preko ESP8266 modula (master na I2C vezi)

```
#include <Wire.h> //Biblioteka koja sadrži skup funkcija za I2C vezu
#include <string.h> //Biblioteka koja sadrži skup funkcija za
manipulisanje string-ovima

#define WIFI_SSID "Simulator Wifi" //SSID WiFi veze koja je dostupna u
simulatoru
#define WIFI_PASSWORD ""
#define HOST "https://iot-ispit-python-webserver.herokuapp.com"
#define PORT 80
#define OTHER_I2C_ID 1 //ID drugog uređaja (slave na I2C komunikaciji)

struct WireData { //Struktura koju koristimo prilikom slanja podataka
        char id='W'; //Oznaka da je poslao WiFiArduino
        char data[30] = "";
};
WireData wireData_from, wireData_to; //Promenljive od kojih je jedna
fiksna (wireData_to), a u drugoj se smešta poruka od slave Arduina
(wireData_from)
bool serverIsBusy = false; //Flag koji signalizira da li server trenutno
šalje podatke preko WiFi modula (stopira redovan rad ovog Arduina u
loop petlji)

void sendDataToServer(String data){ //Funkcija koja šalje podatke na
server

    Serial.print("AT+CIPSEND="); //Slanje započinje ovom komandom...
    Serial.println(data.length()); //...nakon koje odmah sledi dužina
HTTP podatka,...

    Serial.print(data); //...i na kraju i sam podatak
    if(strstr(data.c_str(), "/getCommand") != NULL){ //Ukoliko smo
poslali "getCommand", tj. zahtev koji treba da nam vrati zadatu komandu
za arduino izvršićemo sledeće:
      String httpResponse = Serial.readString(); //Prvo očitavamo HTTP
zahtev sa serijske veze
      Serial.print(""); //*Linija koda iznad neće pravilno očitati bez
ove linije koda
      if(strstr(httpResponse.c_str(), "Content-Length: 0") == NULL)
//Ako ima komande, tj. vraćeni sadržaj nema dužinu 0
        sendCommandFromServer(httpResponse.c_str()); //Prosleđujemo
dobijeni string na dalju obradu kao pokazivač na niz karaktera
      /* Primer dobijenog HTTP odgovora
        "+IPD,93:HTTP/1.1 200 OK //+IPD je oznaka odgovora ESP8266
modula koji je praćen njegovom dužinom i samim odgovorom
        Content-Length: 14
        Content-Type: text/html; charset=utf-8

        |openCloseDoors| //ili je |$changeVentilationSpeed$255$| ili je
|relaySwitch|

        00 OK"
      */
```

```
      else serverIsBusy = false; //Nakon izvršenja funkcije iznad
postavljamo flag na false
    }
    else serverIsBusy = false; //U ovom slučaju su samo poslati podaci
i ne očekujemo odgovor pa odmah postavljamo flag
}

void sendCommandFromServer(const char* data){ //Funkcija koja obrađuje
dobijeni HTTP zahtev i prosleđuje drugom Arduinu
    char dataArray[strlen(data)]; //Pravimo niz karaktera dužine
dobijenog http zahteva
    strcpy(dataArray, data);  //

    char* command; //Ovde smeštamo komandu nakon obrade data niza
karaktera

    const char divider[2] = "|"; //Inicijalizujemo karakter koji deli
ostatak zahteva od komande

    command = strtok(dataArray, divider); //Prvo izvlačimo sve do prvog
pojavljivanja istog

    command = strtok(NULL, divider); //A onda izvlačimo sve ostalo do
drugog pojavljivanja (što i jeste sama komanda)

       Wire.beginTransmission(OTHER_I2C_ID); //Otpočinjemo master ->
slave slanje komande
    String(command).toCharArray(wireData_to.data, String(command).length()
+ 1);

    Wire.write((byte *)&wireData_to, sizeof(WireData)); //Šaljemo
wireData_to podatak kao niz bajtova (veličine naše strukture WireData)
    Wire.endTransmission(); //Završavamo slanje
    delay(1000); //Delay postoji da bi se izvršila komanda na drugom
Arduinu
    sendDataToServer("GET /clearCommand HTTP/1.1\r\nHost: " +
String(HOST) + "\r\n\r\n"); //Slanje komande za brisanje iste iz bafera
na Web serveru nakon što smo je uspešno pročitali
}

void setup(){
    Serial.begin(115200); //Inicijalno ESP8266 modul radi sa baud rate-
om 115200
    Serial.println("AT+UART_DEF=9600,8,1,0,0"); //Kako je baud rate
serijskog porta Arduina uobičajeno 9600, menjamo isti ESP8266 modula.
    //Potpis metoda
AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
Više informacija: https://stackoverflow.com/questions/34248581/set-
baud-rate-to-esp8266-at-9600

    delay(10); //Pravimo kratak delay da bi se izvršila gore navedena
komanda...

    Serial.begin(9600); //...a nakon toga menjamo baud rate serijskog
porta na 9600
    Wire.begin();
```

```
    Serial.println("AT+CIPMODE=1");   // Biramo TCPIP vezu u
transparentnom modu (da bi videli HTTP response)

    do {
        Serial.println("AT+CWJAP=\"" + String(WIFI_SSID) + "\",\"" +
String(WIFI_PASSWORD) + "\""); //Slanje komande za povezivanje na Wifi
konekciju
        //Detaljnija provera odgovora je zakomentarisana iz razloga što
je WiFi modul povezan direktno na Serijski port
        /*if(Serial.find("ERROR")) Serial.println("ERROR OCCURED WHILE
CONNECTION TO WIFI... RETRYING");
        else if(Serial.find("OK")) isWifiConnected = true;
        else Serial.println("?: AT+CWJAP RETURNED \"" +
Serial.readString() + "\"... RETRYING");*/
    } while(!Serial.find("OK")); //Komandu šaljemo sve dok ne dobijemo
pozitivan odgovor da je veza uspostavljena

    do {
       Serial.println("AT+CIPSTART=\"TCP\",\"" + String(HOST) + "\"," +
String(PORT)); //Slanje komande za uspostavljanje TCP konekcije sa
Python serverom
        /*if(Serial.find("ERROR")) Serial.println("ERROR OCCURED WHILE
CONNECTING TO PYTHON SERVER... RETRYING");
        else if(Serial.find("OK")) isTCPConnectionEstablished = true;
        else Serial.println("?: AT+CIPSTART RETURNED \"" +
Serial.readString() + "\"... RETRYING");*/
    } while(!Serial.find("OK")); //Komandu šaljemo sve dok ne dobijemo
pozitivan odgovor da je TCP veza uspostavljena

    //Serial.println("WiFi successfully connected with host: " +
String(HOST) + " on port " + String(PORT));
}

void loop(){
    while(serverIsBusy); //Čekanje dokle god je arduino zauzet, tj.
dokle god komuniciramo sa serverom

    if (Wire.requestFrom(OTHER_I2C_ID, sizeof(WireData))){ //Ukoliko
dobijemo I2C poruku od drugog arduina, mi je obrađujemo
      serverIsBusy = true;

      Wire.readBytes((byte*)&wireData_from, sizeof(WireData));

      if(wireData_from.id != wireData_to.id &&
String(wireData_from.data).charAt(0) == '/') //Dodatna provera da li je
poruka ispravna, a ako jeste onda šaljemo istu na server
        sendDataToServer("GET " + String(wireData_from.data) + "
HTTP/1.1\r\nHost: " + String(HOST) + "\r\n\r\n");
      else //Ukoliko nije, onda znači da je to bila neka druga poruka
pa mi šaljemo getCommand zahtev
        sendDataToServer("GET /getCommand HTTP/1.1\r\nHost: " +
String(HOST) + "\r\n\r\n");
    }
}
```

Kod Arduina koji upravlja celokupnim sistemom i očitava senzore (slave na I2C vezi)

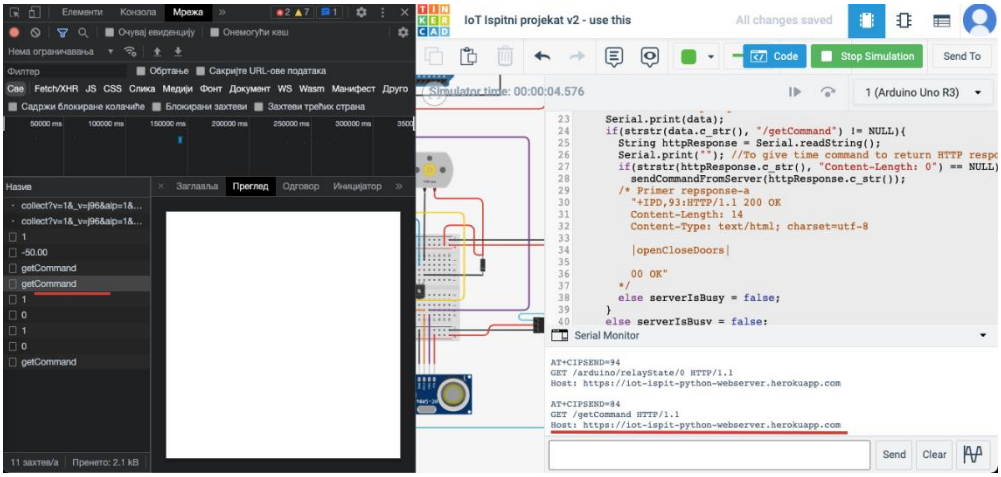
```
#include <Wire.h> //Biblioteka koja sadrži skup funkcija za I2C vezu
#include <Servo.h> //Biblioteka koja sadrži skup funkcija za upravljanje servo motorom

#define SLIDE_SWITCH 2
#define RELAY 4
#define VENTILATION_CONTROL_ANALOG A0
#define VENTILATION 3
#define ULTRASONIC_SENSOR_TRIG 5
#define ULTRASONIC_SENSOR_ECHO 6
#define TEMP_SENSOR_ANALOG A2
#define PHOTORESISTOR_ANALOG A3
#define PHOTORESISTOR_VIN 5
#define PHOTORESISTOR_R 220
#define DOORS_ANALOG A1
#define SERVO_MIN_DEGREE 0
#define SERVO_MAX_DEGREE 90
#define SERVO_MOVE_SPEED 10 //Vrednost mora biti između SERVO_MIN_DEGREE i SERVO_MAX_DEGREE

int previousPotValue, ventilationPower;
bool isDoorOpened, isVentilationUpdatedFromServer;
Servo servoDoor;
unsigned long updateTime;
String wifiResponse = "", wifiSendBuffer = "|";
struct  WireData { //Struktura koju koristimo prilikom slanja podataka
        char id='P'; //Oznaka da je poslao GlavniArduino_1
        char data[30] = "";
};
WireData wireData_from, wireData_to; //Promenljive od kojih je jedna fiksna (wireData_to), a u drugoj se smešta poruka od master Arduina (wireData_from)
const char divider[2] = "|"; //Razdelnik koji razvaja individualne poruke u baferu

void changeRelayState() { //Funkcija koja menja stanje releja
  digitalWrite(RELAY, !digitalRead(RELAY));
  wifiSendBuffer.concat("/arduino/relayState/" +
String(digitalRead(RELAY) == HIGH));
  wifiSendBuffer.concat("|");
}

float readDistanceCM() { //Funkcija koja vraća daljinu objekta na osnovu očitavanja ultrazvučnog senzora
  digitalWrite(ULTRASONIC_SENSOR_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(ULTRASONIC_SENSOR_TRIG, LOW);

  int value = pulseIn(ULTRASONIC_SENSOR_ECHO, HIGH);

  return value / 58; //Konverzija u centimetre
}
```

```
void changeDoor(bool isOpened){ //Funkcija koja otvara/zatvara vrata, tj. menja položaj servo motora
  for(int i = isOpened ? SERVO_MAX_DEGREE : SERVO_MIN_DEGREE; isOpened ? (i - SERVO_MOVE_SPEED) >= SERVO_MIN_DEGREE - SERVO_MOVE_SPEED : (i + SERVO_MOVE_SPEED) <= SERVO_MAX_DEGREE + SERVO_MOVE_SPEED; i = isOpened ? (i - SERVO_MOVE_SPEED) : (i + SERVO_MOVE_SPEED))
  {
    servoDoor.write(i);
    delay(SERVO_MOVE_SPEED < 10 ? 0 : 80); //Neophodan delay u slučaju velike promene ugla servo motora, tj. brzine
  }
}

int photoresistorToLux(int photoresistorValue){ //Funkcija koja pomoću formule vraća očitavanje fotootpornika u lx
  float Vout = float(photoresistorValue) * (PHOTORESISTOR_VIN / float(1023)); // Konverzija iz analognog ulaza u napon
  float RLDR = (PHOTORESISTOR_R * (PHOTORESISTOR_VIN - Vout))/Vout; // Konverzija iz tog napona u otpor
  int lux = 500/(RLDR/1000); // Konverzija otpora u lx
  return lux;
}

void handleServerResponse(const char* command){ //Funkcija koja izvršava zadatu komandu dobijenu od web servera
   char commandArray[strlen(command)];
   strcpy(commandArray, command);

   char* ventialtionValue;
   if(strstr(command, "relaySwitch")!= NULL) changeRelayState(); //Ukoliko je komanda za promene stanja releja
   else if(strstr(command, "$changeVentilationSpeed$")!= NULL){ //Ukoliko je komanda za promenu brzine ventilatora
      const char divider[2] = "$"; //Obrađujemo istu da bi izvukli samo brojčanu vrednost nove brzine

      ventialtionValue = strtok(commandArray, divider);
      ventialtionValue = strtok(NULL, divider);



       command = strtok(NULL, divider);

      if (String(command).toInt() >= 0 && String(command).toInt() <= 255){
          isVentilationUpdatedFromServer = true;
          ventilationPower = String(command).toInt();
          analogWrite(VENTILATION, ventilationPower);
      } else Serial.println("Invalid ventilation speed value -> " + String(command).toInt()); // Neće se ovo nikada desiti jer se vrednost proverava na serveru
  } else if(strstr(command, "openCloseDoors")!= NULL){ //Ukoliko je komanda za otvaranje/zatvaranje vrata
      changeDoor(isDoorOpened);
       isDoorOpened = !isDoorOpened;
```

```
      wifiSendBuffer.concat("/arduino/doorState/" +
String(isDoorOpened));
       wifiSendBuffer.concat("|");
  } else Serial.println("Nepoznata komanda dobijena od strane
servera!"); //U bilo kom drugom slučaju ispisujemo da je nepoznata
komanda
}

void receiveWireData(int size){ //Funkcija koja prima poruku od master
Arduina preko I2C komunikacije
  if(size==sizeof(WireData)){ //Ukoliko je po ispravne dužine onda je
obrađujemo
    Wire.readBytes((byte*)&wireData_from, sizeof(WireData));

    if(wireData_from.id != wireData_to.id){
      //Serial.println(wireData_from.data);
      handleServerResponse(String(wireData_from.data).c_str());
    }
  }
}

void sendWireData(){ //Funkcija koja šalje podatke za server (kako je
šalje se određene dužine koliki je bafer - 30 karaktera, da ne bi
veličina podatka prešla 32 bajta, koliko je dozvoljeno za I2C vezu)
  if(wifiSendBuffer.length() == 0) return;
  char commandArray[wifiSendBuffer.length()];
  strcpy(commandArray, wifiSendBuffer.c_str());
  char* buffer;
  buffer = strtok(commandArray, divider);
  //Serial.println("Sent -> " + String(buffer));
  String(buffer).toCharArray(wireData_to.data, 30);
  Wire.write((byte*) &wireData_to, sizeof(wireData_to));
  wifiSendBuffer.remove(0,String(buffer).length()+1);
}

void setup() {
  Serial.begin(9600);
  Wire.begin(1);

  Wire.onReceive(receiveWireData);
  Wire.onRequest(sendWireData);

  pinMode(SLIDE_SWITCH, INPUT_PULLUP);
  pinMode(RELAY, OUTPUT);
  pinMode(VENTILATION, OUTPUT);
  pinMode(ULTRASONIC_SENSOR_TRIG, OUTPUT);
  pinMode(ULTRASONIC_SENSOR_ECHO, INPUT);

  digitalWrite(RELAY, LOW);
  digitalWrite(VENTILATION, 0);

  attachInterrupt(digitalPinToInterrupt(SLIDE_SWITCH),
changeRelayState, CHANGE);

  ventilationPower = 0;
  servoDoor.attach(DOORS_ANALOG);
  servoDoor.write(0);
```

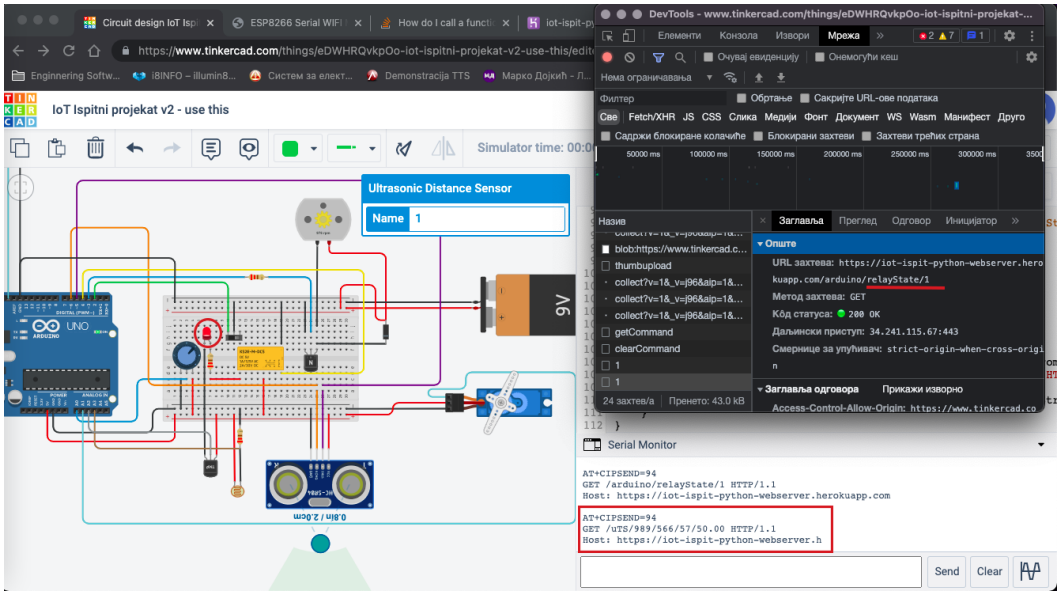
```
  pinMode(TEMP_SENSOR_ANALOG,INPUT);
  pinMode(PHOTORESISTOR_ANALOG,INPUT);

  updateTime = millis() + 1000UL; //Prvo slanje podataka sa senzora ide
nakon uspostave veze sa ESP8266 modulom
}

void loop() {

  if (previousPotValue != analogRead(VENTILATION_CONTROL_ANALOG)) {
//Ukoliko je promenjena vrednost na potenciometru menjamo brzinu DC
motora
    isVentilationUpdatedFromServer = false; //Postavljanje flega na
false da bi mogli kasnije da promenimo brzinu digitalno sa servera
    previousPotValue = analogRead(VENTILATION_CONTROL_ANALOG);
  }

  if(ventilationPower != analogRead(VENTILATION))
analogWrite(VENTILATION, ventilationPower); //Postavljanje brzine DC
motora na zadatu, ukoliko ona već nije
  ventilationPower = isVentilationUpdatedFromServer ? ventilationPower
: map(analogRead(VENTILATION_CONTROL_ANALOG), 0, 1023, 0, 255);
//Postavljanje brzine ventilacije na vrednost potenciometra ukoliko
nije došlo do postavljanje od strane servera

  if((readDistanceCM() < 5 && !isDoorOpened) || (readDistanceCM() >= 5
&& isDoorOpened)){
    changeDoor(isDoorOpened);
    isDoorOpened = !isDoorOpened;

    wifiSendBuffer.concat("/arduino/doorState/" + String(isDoorOpened));
    wifiSendBuffer.concat("|");
  }

  if (millis() >= updateTime)
  {
    wifiSendBuffer.concat("/uTS/" + String(map(ventilationPower, 0, 255,
0, 1502)) + "/" + photoresistorToLux(analogRead(PHOTORESISTOR_ANALOG))
+ "/" +
map(photoresistorToLux(analogRead(PHOTORESISTOR_ANALOG)),2,988,0,100) +
"/" + ((analogRead(TEMP_SENSOR_ANALOG) * 5/1024 - 0.5) * 100));
    wifiSendBuffer.concat("|");

    updateTime = millis() + 1000UL;//60000UL; //Svako sledeće slanje
očitavanja sa senzora se dešava nakon jednog minuta *zbog brzine
simulatora radi demonstracije je stavljeno na 1 sekudnu
  }
}
```

3. Prikaz i objašnjenje rada sistema



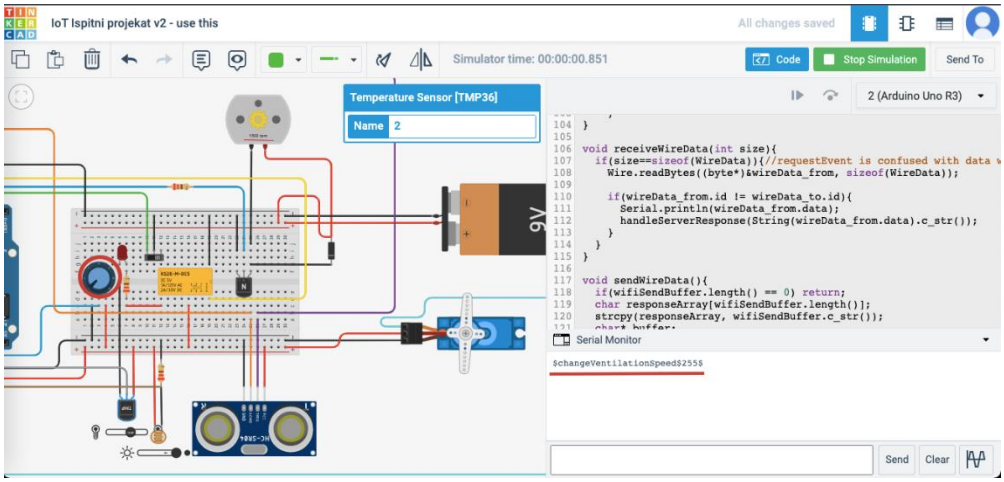
Slika 4:Prikaz stanja sistema kada je sa servera primljena prazna poruka

Na slici 4 vidimo da se prilikom čitanja vrednosti sa servera dobio prazan HTTP odgovor, tj. nije prosleđena nikakva komanda za Arduino.



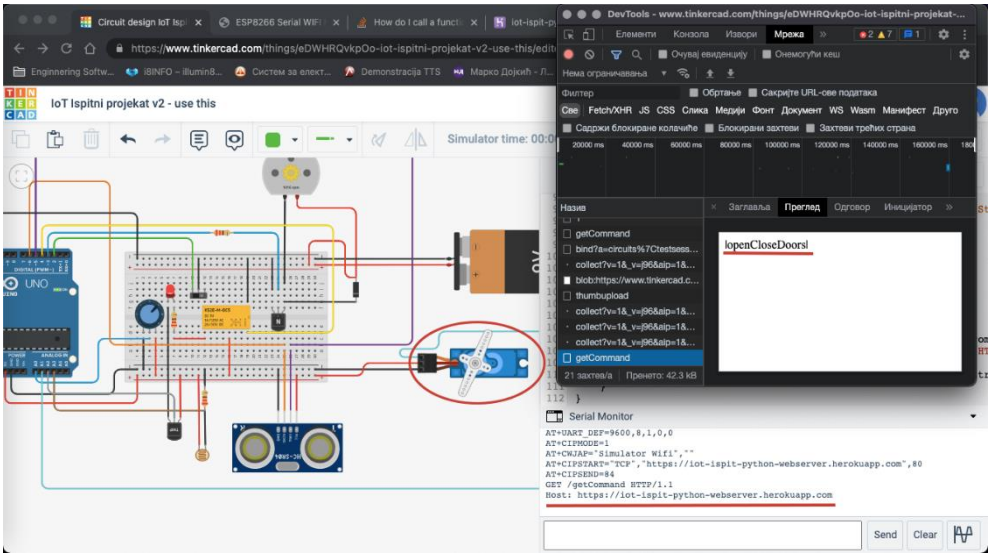
Slike 5:Prikaz stanja sistema kada je poslata informacija o promeni releja na stanje 1, tj. uključena dioda

Na slici 5 vidimo da je uspešno na server poslata informacija o novom stanju releja, tj. 1.



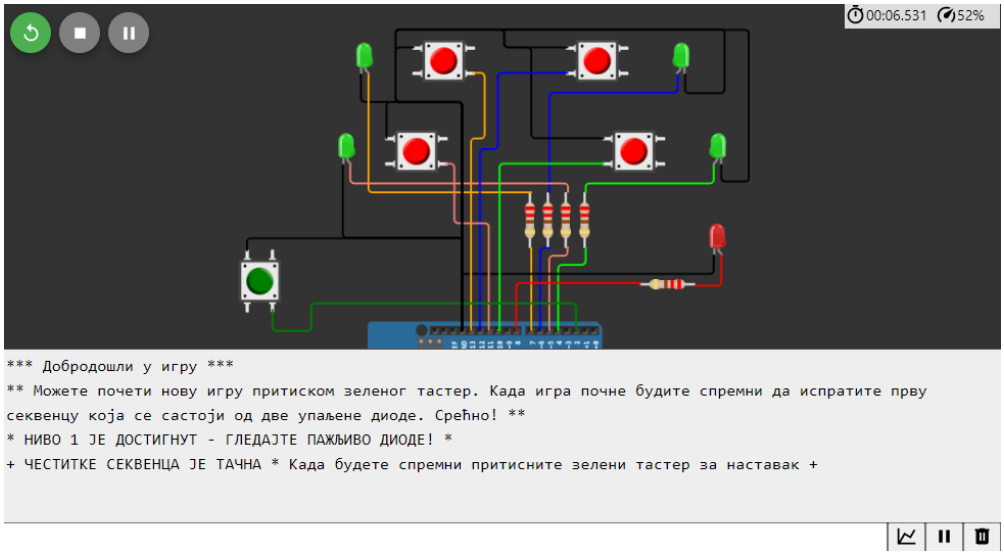
Slike 6:Prikaz stanja sistema kada je primljena komanda za promenu brzine DC motor na 255, tj. maksimalnu

Na slici 6 vidimo kako izgleda komanda za promenu brzine DC motora I da je ona uspešno primljena i primenjena, što dokazuje činjenica da je broj obrtaja motora 1502 RPM, a potenciometar je okrenut na minimum, tj. nulu.



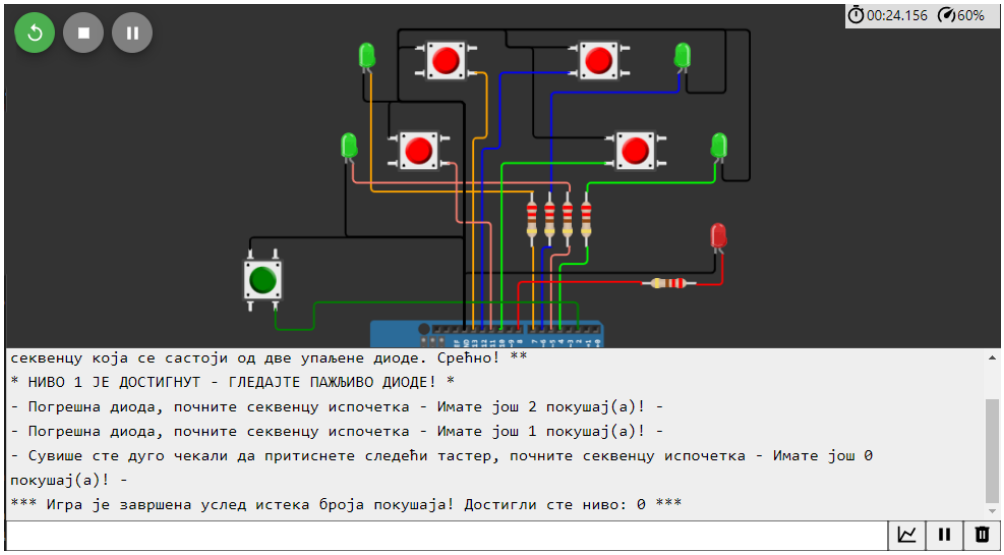
*Slike 7:Prikaz stanja sistema kada je primljena komanda **otvaranje**/zatvaranje vrata, a potom su i poslati podaci sa senzora*

Na slici 7 vidimo da je uspešno primljena komanda za otvaranje vrata (znamo da se radi o otvaranju vrata jer servo motor se pomera ka uglu od 90 stepeni). Takođe na serijskom monitoru vidimo i da je na server poslato redovno očitavanje senzora.



Slika 6:Prikaz stanja sistema pre prelaska na sledeći nivo

Nakon uspešnog ponavljanja sekvence od strane korisnika sistem prelazi u stanje pre prelaska na sledeći nivo. U tom stanju se samo ispisuje poruka čestitke i dodeljuje nova funkcija eksternog prekida zelenom tasteru. Pritiskom na njega, ta funkcija će se i aktivirati, a ona će resetovati brojače koji označavaju broj pokušaja i poslednje pogođenu diodu u sekvenci i promeniti stanje sistema u stanje početka novog nivoa.



Slika 7:Prikaz stanja sistema nakon završetka igre usled isteka broja pokušaja – krajnje stanje sistema

Na slici 7 vidimo, na osnovu poruka serijske veze, da je korisnik stigao do drugog nivoa, a potom usled dovođenja sistema u stanje greške 3 puta (dva puta jer je pogrešio diodu, a treći put jer je čekao suviše dugo da pritisne sledeći taster) doveo sistem u stanje završetka igre.

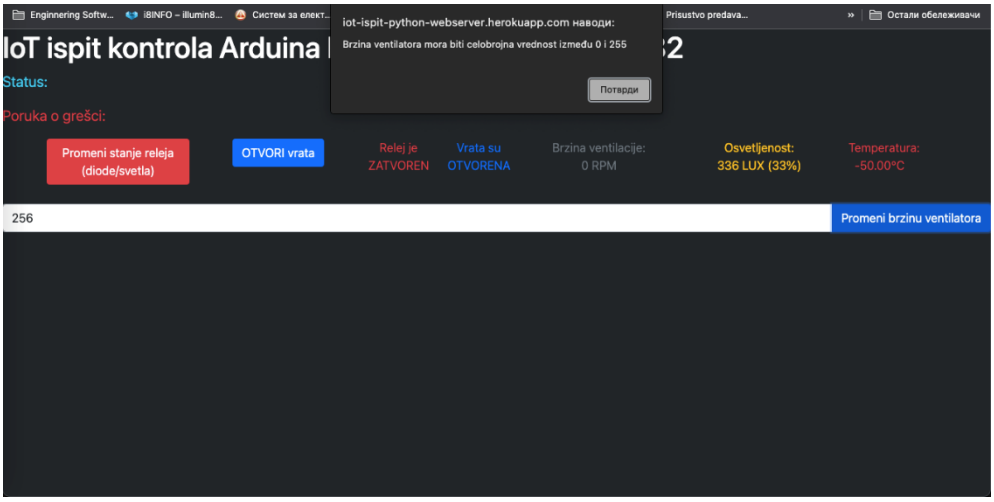
4. Python Flask web server

```
import json
import threading
import time
from flask import Flask, render_template
from turbo_flask import Turbo
from flask_cors import CORS
import requests
from datetime import datetime, timedelta
from flask_mail import Mail, Message
from apscheduler.schedulers.background import BackgroundScheduler


app = Flask(__name__)
turbo = Turbo(app)
#Mail config

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'castleville.crowns@gmail.com'
app.config['MAIL_PASSWORD'] = "ebnh gwbg vurt elun" #Posebna lozinka
samo za ovu aplikaciju
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True

CORS(app)
cors = CORS(app, resource={
    r"/*":{
        "origins":"*"
    }
})

command=""
status=""
errorMessage=""
relayStatus_display = ""
doorStatus_display = ""
ventilation_display = ""
illuminationLux_display = ""
illuminationPercentage_display  = ""
tempCelsius_display = ""

@app.before_first_request
def before_first_request():
    threading.Thread(target=updateFrontend).start() #Nit koja ažurira
html prikaz svih podataka

def updateFrontend():
    with app.app_context():
        while True:
            time.sleep(1)

turbo.push(turbo.replace(render_template("dynamicData.html",
status=status, errorMessage=errorMessage, relayStatus_display =
relayStatus_display, doorStatus_display = doorStatus_display,
```

```
ventilation_display = ventilation_display, illuminationLux_display =
illuminationLux_display, illuminationPercentage_display =
illuminationPercentage_display, tempCelsius_display =
tempCelsius_display),"turboFlaskTarget"))

@app.route('/')
def dashboard():
    global status, errorMessage
    return render_template("dashboard.html")

@app.route('/getCommand', methods=['GET']) #Ruta sa koje se dobija
trenutna komanda za Arduino
def getCommand():
    global command

    app.response_class(
        response=json.dumps(command),
        status=200,
        mimetype='application/json'
    )

    return command, 200

@app.route('/clearCommand', methods=['GET']) #Ruta kojom se ista briše
nakon registrovanja
def clearCommand():
    global command
    command = ""
    return '', 200

@app.route('/changeRelay', methods=['GET']) #Ruta koja se poziva kada
želimo da promenimo stanje releja
def changeDiode():
    global command, status, errorMessage
    if(command == ""):
        command = "|relaySwitch|"
        status = "Promena stanja releja diode/svetla u toku!"
        errorMessage = ""
    else:
        errorMessage = "Već je poslat neki zahtev. Nemoguće je poslati
drugi zahtev!"
    return '', 200

@app.route('/changeVentilationSpeed/<speed>', methods=['GET']) #Ruta
koja se poziva kada želimo da promenimo brzine ventilatora
def changeVentialtionSpeed(speed):
    global command, status, errorMessage
    if(command == ""):
        command = "|$changeVentilationSpeed$" + speed + "$|"
        status = "Promena brzine ventilatora na " + speed + " u toku!"
        errorMessage = ""
    else:
        errorMessage = "Već je poslat neki zahtev. Nemoguće je poslati
drugi zahtev!"
    return '', 200
```

```
@app.route('/openCloseDoors', methods=['GET']) #Ruta koja se poziva
kada želimo da otvorimo/zatvorimo vrata
def openCloseDoors():
    global command, status, errorMessage
    if(command == ""):
        command = "|openCloseDoors|"
        status = "Otvaranje" if relayStatus_display != "1" else
"Zatvaranje" + " vrata u toku!"
        errorMessage = ""
    else:
        errorMessage = "Već je poslat neki zahtev. Nemoguće je poslati
drugi zahtev!"
    return '', 200

@app.route('/sendEmail', methods=['GET']) #Ruta koja se poziva jednom
dnevno za slanje izveštaja putem mejla
    mail = Mail(app)
    subject = "Izveštaj o stanju IoT sistema -
{}".format((datetime.now()- timedelta(days=1)).strftime("%d.%m.%Y"))

    thingspeakData =
requests.get("https://api.thingspeak.com/channels/1726262/feeds.json?re
sults") #Dobijanje svih očitavanja sa Thinkspeak platforme
    feedEntriesJSON = thingspeakData.json()["feeds"]

    temperatureValues = []
    illuminationLuxValues = []
    doorOpenedCount = 0
    relayStateChangeCount = 0
    previousRelayState = -1

    for feedEntry in feedEntriesJSON:
        if(datetime.strptime(feedEntry["created_at"], "%Y-%m-
%dT%H:%M:%SZ").strftime("%Y-%m-%d") == (datetime.now()-
timedelta(days=1)).strftime("%Y-%m-%d")): #Filtriramo samo one koji su
od predhodnog dana
            if(feedEntry["field6"] != None):
temperatureValues.append(float(feedEntry["field6"]))
            if(feedEntry["field4"] != None):
illuminationLuxValues.append(float(feedEntry["field4"]))
            if(feedEntry["field3"] != None and int(feedEntry["field3"])
== 1): doorOpenedCount += 1
            if(feedEntry["field1"] != None and int(feedEntry["field1"])
!= previousRelayState):
                previousRelayState = int(feedEntry["field1"])
                relayStateChangeCount += 1

    body = "\
        IoT izveštaj za dan {}\n\
        Prosečna vrednost temperature: {} °C\n\
        Prosečna osvetljenost senzora: {} Lux\n\
        Ukupan broj otvaranja vrata: {}\n\
        Ukupan broj promena stanja na releju diode/svetla: {}\
    ".format((datetime.now()- timedelta(days=1)).strftime("%d.%m.%Y"),
        round(sum(temperatureValues)/len(temperatureValues), 2) if
len(temperatureValues) > 0 else 0.0,
```

```
        round(sum(illuminationLuxValues)/len(illuminationLuxValues), 2)
if len(temperatureValues) > 0 else 0.0,
        doorOpenedCount, relayStateChangeCount)

    message = Message(subject, sender = 'castleville.crowns@gmail.com',
recipients = ['marko.dojkic.18@singimail.rs'])

    message.body = body

    mail.send(message)
    return '', 200

@app.route('/arduino/relayState/<relayState>', methods=['GET']) #Ruta
koja se okida kada dobijemo informaciju o promeni stanja na releju
def updateRelayState(relayState):
    global relayStatus_display
    relayStatus_display = relayState

requests.get("https://api.thingspeak.com/update?api_key=78YAN1V93W693DP
A&field1=" + relayState)
    return '', 200

@app.route('/arduino/doorState/<doorState>', methods=['GET']) #Ruta
koja se okida kada dobijemo informaciju da su se vrata otvorila odnosno
zatvorila
def updateDoorState(doorState):
    global doorStatus_display
    doorStatus_display = str(doorState)

requests.get("https://api.thingspeak.com/update?api_key=78YAN1V93W693DP
A&field3=" + doorState)
    return '', 200

@app.route('/uTS/<ventilation>/<illuminationLux>/<illuminationPercentag
e>/<tempCelsius>', methods=['GET']) #Ruta koja se okida kada dobijemo
periodična očitavanja sa senzora (svaki minut)
def
updateReadings(ventilation,illuminationLux,illuminationPercentage,tempCel
sius):
    global ventilation_display, illuminationLux_display,
illuminationPercentage_display, tempCelsius_display
    ventilation_display = ventilation + " RPM"
    illuminationLux_display = illuminationLux + " LUX"
    illuminationPercentage_display = illuminationPercentage + "%"
    tempCelsius_display = tempCelsius + "°C"

requests.get("https://api.thingspeak.com/update?api_key=78YAN1V93W693DP
A&field2=" + ventilation + "&field4=" + illuminationLux + "&field5=" +
illuminationPercentage + "&field6=" + tempCelsius)
    return '', 200

if __name__ == "__main__":
    mailSendingBackgroundScheduler = BackgroundScheduler()
    mailSendingBackgroundScheduler.add_job(sendEmail, 'interval',
hours=24) #Definisanje posla za slanje mejla u pozadini
    mailSendingBackgroundScheduler.start()
    app.run(port=5000, debug=True)
```



Slika 8: Prikaz izgleda web servera nakon pokušaja slanja ne dozvoljene vrednosti brzine ventilacije tj. DC motora

Na slici 8 vidimo JavaScript alert koji nam ukazuje da smo pokušalji da pošaljemo vrednost za brzinu DC motora koja je van dozvoljenog opsega. Takođe vidimo i poslednja očitavanja sa senzora.

Slika 9: Prikaz izgleda web servera nakon pokušaja slanja nove komande dok predhodna nije ni primljena

Na slici 9 vidimo poruku o grešci koja je nastala jer smo hteli da pošaljemo novu komandu na Arduino, a vidimo da je trenutni status **Otvaranje** vrata. Takođe vidimo i nova očitavanja.

Slika 10: Prikaz izgleda mejla izveštaja na dan 8.5.2022.