

Sveučilište u Splitu
Fakultet elektrotehnike, strojarstva i brodogradnje

SEMINARSKI RAD

Sustav za održavanje stanice baze podataka

Josip Krišto

Split, svibanj 2017.

Sadržaj:

1. Uvod	3
2. Korištena oprema	4
2.1. MotorBee pločica	4
2.2. StepperBee pločica	5
2.3. Wasp pločica	6
2.4. Senzor zvuka	7
2.5. Senzor svjetlosti	8
3. Hardware	9
4. Software	11
5. Zaključak	14
6. Dodatak	15

1. Uvod

U ovom seminarskom radu biti će opisana realizacija sustava za održavanje stanice baze podataka tj. hlađenje iste i održavanje određene temperature u stanici te periodičko vođenje evidencije o trenutnim temperaturama i njezino grafičko prikazivanje te mogućnost pregleda temperatura u bilo kojem prošlom trenutku rada stanice na tekstualni i grafički način.

Ovaj sustav je realiziran MotorBee i StepperBee razvojnim pločicama, WASP pločicom te senzorom zvuka i senzorom svjetlosti koji će u našem slučaju „glumiti“ senzor temperature.

Računalna aplikacija za upravljanje sustavom je napisana u C# programskom jeziku te se sastoji od 3 dijela (forme). Prvi dio koji služi za identifikaciju ovlaštenih osoba pomoću korisničkog imena i lozinke, drugi dio je glavni dio koji služi za automatsko ili ručno upravljanje hlađenjem stanice te grafički i tekstualni prikaz trenutnih i prošlih temperatura i treći dio služi za grafički prikaz temperatura odabranog radnog ciklusa stanice od trenutka njenog paljenja do njenog gašenja.

U nastavku će se opisat hardware-ski te software-ski dio detaljnije te će biti priložen programski kod računalne aplikacije sa priloženim slikama.

2. Korištena oprema

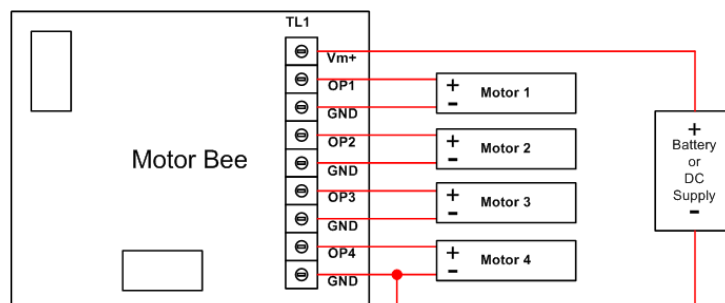
U ovom poglavlju se nalaze korištene komponente za realizaciju sustava, njihov pregled i izgled te primjena u sustavu.

2.1. MotorBee pločica

MotorBee modul je integrirana pločica koja ima 4 digitalna izlaza na koja možemo spojiti razne potrošače kojima namjeravamo upravljati. Pomoću USB porta ovaj se modul jednostavno spaja sa računalom. Pločica sadrži 4 digitalna izlaza (PL3), te 6 digitalnih ulaza (PL2). Za realizaciju našeg sustava koristit ćemo 3 digitalna ulaza za senzor zvuka te ćemo pomoću nje upravljati DC motorom koji ima ulogu ventilatora za hlađenje u sustavu.



Slika 2.1. MotorBee modul



Slika 2.2. Spajanje 4 nezavisna motora na modul

DC motor

DC motor je vrsta rotirajućeg električnog stroja koji pretvara istosmjernu električnu struju u mehanički rad. DC motor korišten u ovom radu ima relativno malu brzinu vrtnje. Potrebno mu je vanjsko napajanje od 6V. Za ispravno djelovanje potreban mu je upravljač kao što je MotorBee modul.

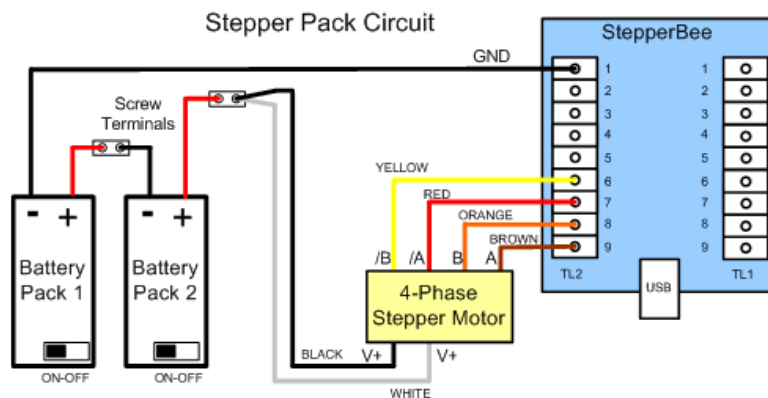
U ovom radu DC motor će služiti, što smo već ranije naveli kao ventilator za hlađenje sustava.



Slika 2.3. DC motor

2.2 StepperBee pločica

StepperBee je također integrirana pločica te je složenije izvedbe od MotorBee pločice zbog različitih mogućnosti upravljanje step motorom koji su dosta precizni te imaju više načina rada. Tj. dok obični DC motori zahtijevaju samo određenu voltažu za rad, step motori trebaju preciznu sekvencu pulseva dostavljenu na ispravan priključak u točno određeno vrijeme da bi ispunili svoj zadatak. I ova pločica ima jedan USB priključak za komunikaciju sa računalom, 5 digitalnih ulaza. Ova pločica je dizajnirana za najčešći tip step motora a to je unipolarni sa 4 faze.



Slika 2.4. Shema spanja stepper motora na StepperBee modul

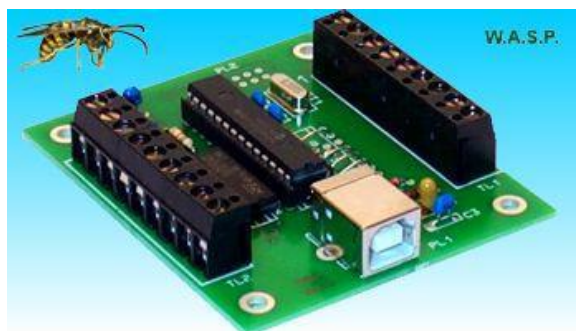
Step motor

U našem slučaju koristimo najčešći tip step motora sa 4 faze i već smo naveli da su step motori dosta precizniji i zahtjevniji od običnih DC motora. Step se za razliku od DC motora može gibati u koracima od samo nekoliko stupnjeva (ovisno o motoru) za što nam je potreban upravljački sklop i to StepperBee modul.

U našem sustavu step motor nema neku posebnu namjenu već je korišten samo kao dodatni ventilator za hlađenje stanice u slučaju prevelikog zagrijavanja.

2.3 WASP pločica

WASP pločica nudi pogodan način spajanja PC-a sa realnim svijetom analognih i digitalnih signala. Ima 4 analoga ulaza i 7 digitalnih izlaza. Analogni ulazi prihvaćaju napon u opsegu od 0-5 V a digitalni izlazi su sposobni uključivati i isključivati širok opseg uređaja direktno. Ova pločica se spaja sa PC preko standardnog USB sučelja i PC će je prepoznati odmah te automatski konfigurirati za rad. Analogni ulazi mogu prihvatiti širok opseg senzora za upravljanje širokim opsegom mogućih aplikacija.



Slika 2.5. Wasp modul

U našem sustavu koristit ćemo senzor svjetlosti koji spajamo na analogne ulaze pločice i koji daje vrijednosti od 0 do 255. Koristit ćemo ga kao zamišljeni senzor temperature tako što ćemo vrijednosti od 0 do 255 pretvoriti u zamišljene vrijednosti temperature od 5 do 65 stupnjeva.

2.4 Senzor zvuka

Kako bi se uočila buka u prostoriji potreban nam je senzor zvuka i u našem slučaju to je SEN002 senzor. Ovaj senzor je vrlo pogodan način za detektiranje zvuka sa ručnim ili automatskim sustavom. Ima osjetljivi mikrofoni i pojačalo te mu se može mijenjati razina osjetljivosti tako da odgovara određenoj primjeni. Izlaz je jednostavno binaran, kada je zvuk iznad postavljene granice izlaz ide u logičku 0 (0V) a kada je zvuk ispod granice izlaz je u logičkoj 1 (+5V). Također ima i ugrađenu ledicu koja svijetli kada zvuk prođe određenu granicu.

Ima 3 konektorska pina: VCC – služi za napajanje +5V DC

GND – uzemljenje 0V

OUT – izlaz digitalnog logičkog signala

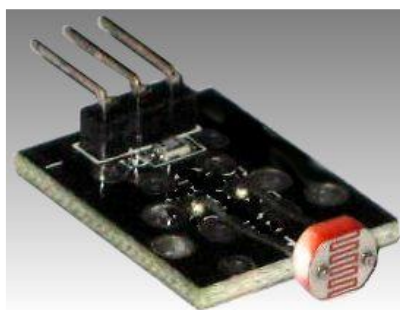


Slika 2.6 Senzor zvuka

2.5 Senzor svjetlosti

Senzor svjetlosti je vrlo jednostavan za koristiti ali vrlo svestran obzirom na bilo koji upravljački i automatizirani sustav. Gdje god je potrebno izmjeriti količinu osvijetljenosti ovaj uređaj može biti savršeno rješenje. Zasniva se na otporniku ovisnom o svjetlu (LDR) koji izviruje sa prednjeg ruba pločice omogućavajući da bude diskretno montiran tako da se samo LDR vidi. Ima 3 konektorska pina i zahtjeva 5V DC za rad. Kako zahtjeva malu struju, vanjski 5V je najčešće pogonjen sa običnim USB kablom ili slično. Izlaz daje analogni napon u rasponu od 0 do +5V proporcionalno intenzitetu detektirane svjetlosti.

U našem sustavu će biti korišten kao „zamišljeni“ senzor temperature tako što ćemo pomoću jednačine pretvarati raspon signala od 0 do 255 u signal od 5 do 65 stupnjeva. Inače se koriste pravi senzori temperature koji su jeftini i dostupni te jako precizni ali mi ih nismo imali na raspolaganju te smo iskoristili ovaj senzor.



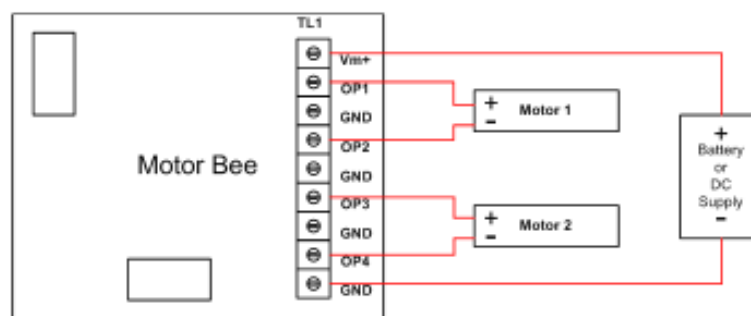
Slika 2.7. Senzor svjetlosti

3. Hardware

Kao što smo već i ranije naveli za realizaciju ovog rada korišten je sljedeći hardware:

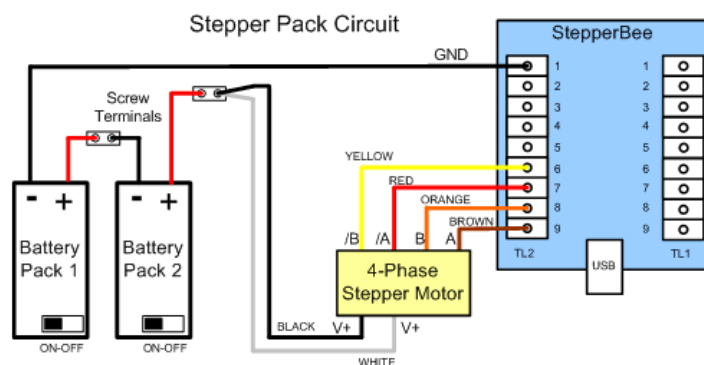
- DC motor i Step motor
- Senzor zvuka i svjetlosti
- WASP modul
- MotorBee i StepperBee moduli
- Računalo
- Vanjsko napajanje i USB kabeli

Ovdje ćemo ukratko opisat kako je sve skupa povezano da bi funkcioniralo te priložiti sheme spajanja motora i senzora na same module od kojih su neke već ranije prikazane. Kao prvo jedna od najbitnijih komponenti je samo računalo pomoću kojega ćemo upravljati svim komponentama. Sva tri modula: MotorBee, StepperBee i WASP su spojena USB kabelom na računalo koje ujedno služi i za napajanje i za komunikaciju. DC motor je spojen na MotorBee modul na pinove OP1 i OP2 tako da smo u mogućnosti da se vrti u oba smjera. Također DC motor zahtjeva vanjsko napajanje od 6V koje ćemo dovesti na pinove Vm i GND MotorBee modula sa ispravljača (vanjsko napajanje).



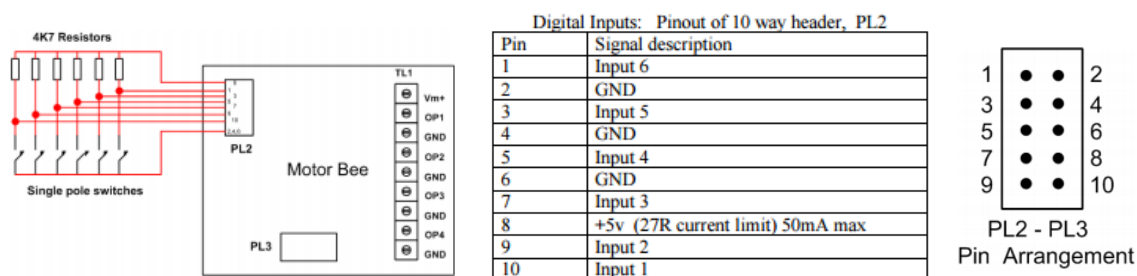
Slika 3.1. Shema spajanja DC motora na MotorBee modul

Step motor se spaja na StepperBee modul na 4 pina što je prikazano na shemama jer je to 4-fazni motor. Također i step motor zahtjeva vanjsko napajanje od 12 V koje ćemo također dovesti na ispravljača na motor i modul.

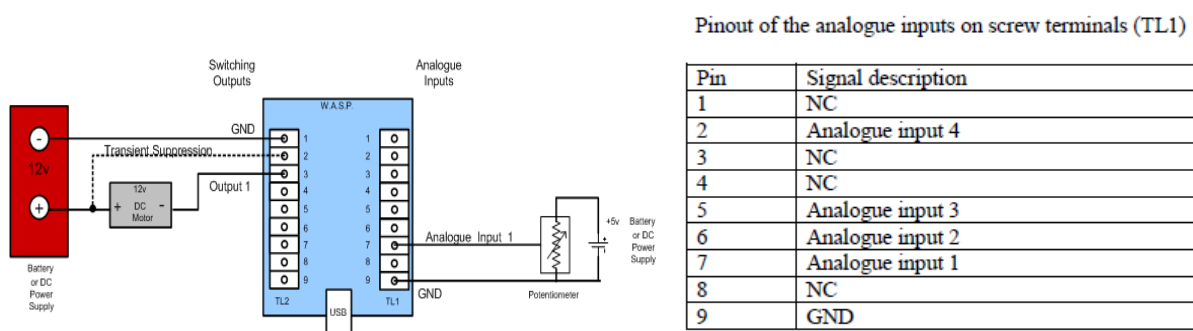


Slika 3.2. Shema spajanja stepper motora na StepperBee modul

Senzor zvuka spajamo na digitalne ulaze PL2 MotorBee modula dok senzor svjetlosti spajamo na potrebni WASP modul na analogne ulaze.



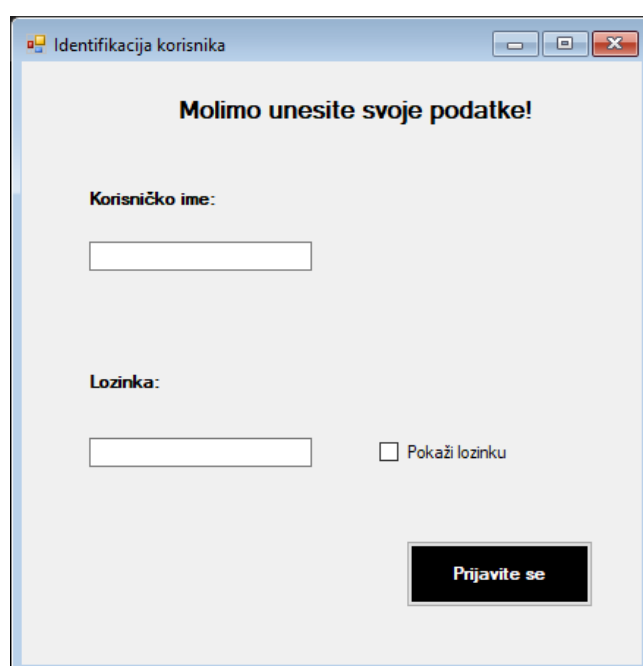
Slika 3.3. Shema spajanja senzora zvuka na digitalni ulaz MotorBee modula (PL2) i raspored te funkcija pojedinih pinova



Slika 3.4. Shema spajanja senzora svjetlosti na analogne ulaze Wasp modula i funkcija pinova

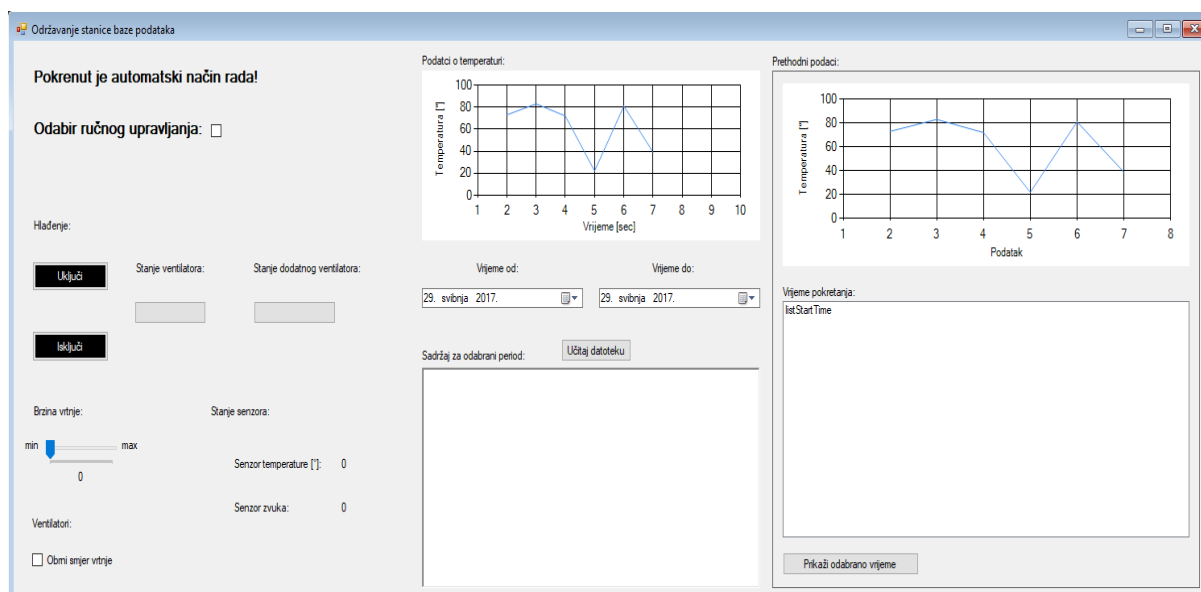
4. Software

Osim hardware potreban je i software odnosno računalna aplikacija koje će upravljati cijelim sustavom. Računalna aplikacije je napravljena u C# programskom jeziku te se sastoji od 2 forme: forme za identifikaciju ovlaštenih osoba i glavne forme za ručno ili automatsko upravljanje održavanju stanice baze podataka. Prvu formu možemo vidjeti na slici ispod.

The image shows a Windows-style application window titled "Identifikacija korisnika". Inside the window, the text "Molimo unesite svoje podatke!" is centered at the top. Below this, there are two labels: "Korisničko ime:" followed by a text input field, and "Lozinka:" followed by another text input field. To the right of the password field is a checkbox labeled "Pokaži lozinku". At the bottom right of the form area is a button labeled "Prijavite se".

Slika 4.1. Izgled forme za identifikaciju korisnika

Ova forma je prilično jednostavna i sastoji se od 2 textboka, 1 checkboxa i 1 bottuna. Kao što se vidi u polja se unose korisnički podaci: ime i sifra te ako je pravilno unešeno moguće je upravljanje sustavom. Imamo checkbox pomoću kojeg možemo vidjeti na trenutak znakove sifre u slučaju da smo pogriješili te obični bottun za pristup glavnoj formi. Ako je identifikacija ispravna otvara nam se glavna forma koju možemo vidjeti na slici ispod.



Slika 4.2. Izgled glavne forme programa sa automatskim i ručnim načinom upravljanja

Ukratko ćemo objasniti glavnu formu. U gornjem lijevom kutu imamo mogućnost „premošćivanja“ programa te odabir ručnog upravljanja sustavom. Kada je odabrano ručno upravljanje imamo mogućnosti paliti i gasiti glavni motor (ventilaciju) te birati željenu brzinu samog ventilatora na trackbaru koji se nalazi ispod. Također možemo i vidjet ispod vrijednost trenutne brzine. Ispod se još nalazi checkbox za odabir suprotnog smjera ventilatora što je korisno s vremena na vrijeme napraviti radi čišćenja prašine.

Zatim pokraj bottuna za paljenje i gašenje ventilatora imamo 2 indikatora stanja glavnog i pomoćnog ventilatora (zeleno ako je upaljen, crveno za ugašen). Ispod se nalaze stanja senzora zvuka i senzora temperature. Ako je uključen automatski mod i ako je temperatura u granicama od 5 do 20 stupnjeva tada se glavni ventilator vrti pri manjoj brzini, ako temperatura pređe 20 stupnjeva i ispod je 40 tada će se ventilator početi okrećati većom brzinom i ako je na kraju temp prešla 40 tada se ventilator vrti najvećom brzinom.

Također imamo senzor zvuka koji ako detektira buku koja je uzrokovana glavnim ventilatorom on će pokrenuti pomoćni ventilator radi boljeg hlađenja.

Na sredini forme imamo graf koji iscrtava u realnom vremenu trenutnu temperaturu te kod koji periodički sprema temperaturu u datoteku temp.txt file sa vremenom kada je ta temperatura zabilježena. Tu još imamo i mogućnost učitavanja samog sadržaja datoteke u željenom rasponu vremena i pogled na prethodno stanje temperature.

I na kraju desno imamo također graf i ispod njega listu svih vremena kada je program (sustav) bio pokrenut. U ovom kodu je urađena serijalizacija podataka u datoteci i deserijalizacija pomoću json framewoka. Ovdje se kreiraju liste u koje se spremaju objekti koji sadržavaju vrijeme i temperaturu. Odabirom jednog od vremena kada je program bio pokrenut učitavaju se temperature zapisane u toj datoteci koja je vezana samo za to vrijeme rada te se one iscrtavaju na gornji graf.

5. Zaključak

Ovim radom smo pokušali riješiti problematiku hlađenje same stanice baze podataka odnosno hlađenja svih komponenti u prostoriji. Sa ovim radom su otvorene daljne mogućnosti rješavanja ove problematike na bolje i drugačije načine. Rad je uspješno napravljen te obavlja svoju zadanu funkciju i osim osnovnih funkcionalnosti imamo i grafičke prikaze trenutnih temperatura i temperatura iz prošlih procesa rada sustava što nam daje odličan uvid u samu kvalitetu sustava hlađenja i lakši pogleda na sve podatke.

Naravno ovaj sustav bi mogao dosta bolje i preciznije obavljati posao da smo koristili kvalitetnije senzore za buku, pravi i dobre senzore temperature te cijeli niz ventilatora sa mogućnosti velike brzine vrtnje.

Sa ovim jednostavnim primjerom realizacije sustava uočavamo da se dosta toga može napraviti iz prve ruke korištenjem povoljne elektroničke opreme uz poznavanje programiranja i vođenja procesa što je jako korisno za vlastiti potrebe u manjim sustavima.

6. Dodatak

U ovim dijelu ćemo priložiti pogramski kod cijele računalne aplikacije sa priloženim komentarima te kratkim pojašnjenjem.

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SeminarskiRad
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SeminarskiRad
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

        this.StartPosition = FormStartPosition.CenterScreen;
        this.AcceptButton = button1;           // postavljanje primarnog dugmeta kada
se pritisne Enter tipka
    }

    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        Application.Exit();           // zatvaranje aplikacije pri zatvorenju forme
    }

    private void izlazToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void button1_Click(object sender, EventArgs e)
// provjera imena i sifre te otvaranje nove forme
    {
        if ((textBox1.Text == "Josip Kristo" && textBox2.Text ==
"lozinka111") || (textBox1.Text == "Marko Bosnjak" && textBox2.Text == "lozinka222"))
        {
            Form3 form3 = new Form3();
            form3.Show();
            this.Hide();

        }
        else

        {
            MessageBox.Show("Pogresan unos podataka!" + "\n" + "Molimo pokusajte
ponovo.");

            textBox2.Text = "";
            textBox1.Focus();

        }
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
// omogućavanje pogleda sifre pri klikom na checkbox
    {
        if (checkBox1.Checked == true)
        {
            textBox2.PasswordChar = '\0';
        }
        else
        {
            if (checkBox1.Checked == false)
            {
                textBox2.PasswordChar = '*';
            }
        }
    }
}

```


Form3.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Runtime.InteropServices;
using Newtonsoft.Json;

namespace SeminarskiRad
{
    public partial class Form3 : Form
    {
        StreamReader citanje;
        StreamWriter pisanje;
        double zamisljena_temp;
        double temperatura;           // pretvaranje vrijednosti 0-255 u
temp raspona 5-65°

        bool pokrenut_ventilator2 = false; // ventilator2 (step motor) u početnom
stanju ne radi
        bool pokrenut_ventilator = true;   // ventilator (dc motor) u početnom
stanju radi

        int brzina_ventilatora1 = 60;
        int brzina_ventilatora12 = 150;
        int brzina_ventilatora123 = 250;
        int zvuk;

        List<TempData> TrenutniPodaci = new List<TempData>(); // Lista koja cuva
vrijednosti svakog TempData objekata.

        string startTime = ""; // vrijeme pokretanja programa koje se koristi za
naziv datoteke

        public Form3()
        {
            InitializeComponent();
            MotorDC.InitMotoBee();           // inicijalizacija DC motora
            MotorStepper.InitStp();          // inicijalizacija step motora
            WASP.InitWasp();                 // inicijalizacija senzora svjetlosti

            this.StartPosition = FormStartPosition.CenterScreen;
        }
    }
}
```

```

private void Form3_Load(object sender, EventArgs e)
{
    MotorDC.SetMotors(1, 30, 0, 0, 0, 0, 0, 0, 0); // pokretanje
    motora(ventilatora) pri otvaranju forme te ukljucivanje "ledice" (bottun)

    if (pokrenut_ventilator )
    {
        pokrenut_ventilator = true;

        button4.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }

    checkBox1.Checked = false; // onemogućavanje
    svih ručnih opcija jer je uključen automatski mod
    checkBox1.Enabled = false;

    button1.Enabled = false;
    button2.Enabled = false;

    trackBar1.Enabled = false;

    startTime = DateTime.Now.ToString("yyyy-MM-dd_HH-mm-ss"); // Vrijeme
    kada je program pokrenut

    DirectoryInfo dinfo = new DirectoryInfo(Application.StartupPath);
    // popuni listStartTime sa dostupnim datotekama od svakog pokretanja
    FileInfo[] Files = dinfo.GetFiles("*.json");

    List<ListBoxItem> lbItems = new List<ListBoxItem>(); //
    Lista koja ce cuvati sve objekte ListBoxItems i na kraju ih dodati u ListBox
    listStartTime
    foreach (FileInfo file in Files)
    {
        string[] FileNamePartsA = file.Name.Replace(".json", "").Split('_');
        // Skloni extenziju iz naziva file i razdvoji ga na datum i vrijeme

        string[] FileDatePart = FileNamePartsA[0].Split('-');
        // razdvoji datum

        string[] FileTimePart = FileNamePartsA[1].Split('-');
        // razdvoji vrijeme

        // dodaj naš objekat ListBoxItem u listu gdje je vrijednost naziv
        filea a text formatirani datum i vrijeme
        lbItems.Add(new ListBoxItem() { Value = file.Name, Text =
        FileDatePart[2] + "." + FileDatePart[1] + "." + FileDatePart[0] + ". " +
        FileTimePart[0] + ":" + FileTimePart[1] + ":" + FileTimePart[2] });
    }

    listStartTime.DisplayMember = "Text";
    listStartTime.ValueMember = "Value";
    listStartTime.DataSource = lbItems;
}

```

```

        // ako zelimo da se file temp.txt isprazni prilikom svakog pokretanja
        programa otkomentirati ispod kod

        /*FileStream datoteka = new FileStream("temp.txt", FileMode.Truncate,
        FileAccess.Write);
        datoteka.Close();*/

    }

private void Form3_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
    // pri zatvaranju forme gasimo aplikaciju te motore

    MotorDC.SetMotors(0, 0, 0, 0, 0, 0, 0, 0, 0);
    MotorStepper.StopMotor1(0);

    pokrenut_ventilator = false;
    pokrenut_ventilator2 = false;

    var json = JsonConvert.SerializeObject(TrenutniPodaci);    //
    Serijalizacija TrenutniPodaci liste koja sadrži objekte TempData i spremanje u
    datoteku sa ekstenzijom .json

    FileStream dataFS = new FileStream(startTime+".json", FileMode.Create,
    FileAccess.Write);
    StreamWriter dataSW = new StreamWriter(dataFS);
    dataSW.Write(json);
    dataSW.Close();
}

private void timer1_Tick(object sender, EventArgs e)           // pokretanje timer1 sa
otvaranjem forme
{
    zvuk = SenzorZvuka.DohvatiInput();

    label11.Text = Convert.ToString(zvuk);                      //
    ispisavanje vrijednosti 0 ili 1 sa senzora zvuka

    temperatura = (0.2353*zamisljena_temp)+5;                  // pretvaranje
    vrijednosti 0-255 u raspon temp. od 5-65
    zamisljena_temp = WASP.ProcitajInput();
    label15.Text = temperatura.ToString();                      // ispisavanje
    vrijednosti temp. u labelu

    if (!checkBox2.Checked)                                     // ako je
    ukljucen automatski mod prati se koja je temperatura te u ovisnosti od odredene temp.
    ventilator se postavlja na odredenu brzinu

    {
        if (temperatura >= 5 && temperatura <= 20)
        {

            pokrenut_ventilator = true;
            MotorDC.SetMotors(1, brzina_ventilatora1, 0, 0, 0, 0, 0, 0, 0);
            trackBar1.Value = brzina_ventilatora1;

```

```

        label12.Text = brzina_ventilatora1.ToString();
        button4.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }

    if(temperatura >= 21 && temperatura <= 40)
    {
        pokrenut_ventilator = true;
        MotorDC.SetMotors(1, brzina_ventilatora12, 0, 0, 0, 0, 0, 0, 0);
        trackBar1.Value = brzina_ventilatora12;
        label12.Text = brzina_ventilatora12.ToString();
        button4.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }
    if (temperatura >= 41 && temperatura <= 65)
    {
        pokrenut_ventilator = true;
        MotorDC.SetMotors(1, brzina_ventilatora123, 0, 0, 0, 0, 0, 0, 0);
        trackBar1.Value = brzina_ventilatora123;
        label12.Text = brzina_ventilatora123.ToString();
        button4.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }

}

}

private void timer2_Tick(object sender, EventArgs e) // drugi timer koji
sluzi za zapisivanje temp. u datoteku cijelo vrijeme svakih 5 sek.
{
    string temp_zapis = Convert.ToString(temperatura);

    string datum = DateTime.Now.ToString(); // spremanje
trenutnog datuma i temp u string
    string temp_datum = datum + " " + temp_zapis + " °"; // spremanje
datuma i temp. sa razmakom u string

// ne
koristimo globalni filestream jer onda ne bi znali jeli otvoren već prije jer ćemo u
isto vrijeme i pisati i čitati te bi se javile greške

    FileStream datotekaWR = new FileStream("temp.txt", FileMode.Append,
FileAccess.Write); // Pri svakom ticku timera otvara se novi filestream ali u
Append modu jer Open raadi truncate pa u fileu uvijek ima samo jedna linija zapisana.
    pisanje = new StreamWriter(datotekaWR);
    pisanje.WriteLine(temp_datum);
    pisanje.Close(); // zatvaranje streamwritera da
bi se sadržaj nasao u fileu te se zatvara i filestream pa se pri idućem prolazu ponovo
otvara

    TrenutniPodaci.Add(new TempData() { datum = datum, temperatura =
temp_zapis }); // Dodavanje objekta TempData u listu TrenutniPodaci svakih 5 sek
}

private void button3_Click(object sender, EventArgs e)
{

```

```

        richTextBox1.Clear(); // brišemo
prethodni sadržaj box-a

        DateTime datumOd = dateTimePicker1.Value.Date;
        DateTime datumDo = dateTimePicker2.Value.Date;

        String linija;

        FileStream datotekaRD = new FileStream("temp.txt", FileMode.Open,
        FileAccess.Read); // otvaranje novog filestream-a da bi citanje radilo
        citanje = new StreamReader(datotekaRD);

        while (!citiranje.EndOfStream) // citanje svih linija
        datoteke te usporedivanje datuma sa odabranim i zapisivanje u richtextbox
        {
            linija = citanje.ReadLine();
            String[] niz = linija.Split();
            DateTime datum = DateTime.Parse(niz[0]);

            if ((datum >= datumOd) && (datum <= datumDo))
            {
                richTextBox1.Text += linija + "\n";
            }
        }

        citanje.Close(); // zatvaramo streamreader a samim tim i filestream koji
je korišten
    }

    private void checkBox2_CheckedChanged(object sender, EventArgs e)
    {
        if (checkBox2.Checked) // paljenje ručnog
        moda označavanjem checkbox-a te omogućavanje kontrola
        {
            checkBox1.Enabled = true;
            button1.Enabled = true;
            button2.Enabled = true;
            trackBar1.Enabled = true;
        }
        else
        {
            checkBox1.Enabled = false;
            checkBox1.Checked = false;
            button1.Enabled = false;
            button2.Enabled = false;
            trackBar1.Enabled = false;
        }
    }

    private void trackBar1_Scroll(object sender, EventArgs e)
    {
        int x = trackBar1.Value; // Mijenjanje
        brzine ventilatora sa trackbar-om te ispisivanje brzine
        if (pokrenut_ventilator)
        {
            MotorDC.SetMotors(1, x, 0, 0, 0, 0, 0, 0, 0);
        }
    }

```

```

        label12.Text = Convert.ToString(x);

    }

    private void button1_Click(object sender, EventArgs e)
    {
        MotorDC.SetMotors(1, 100, 0, 0, 0, 0, 0, 0, 0);           // paljenje
        ventilatora te postavljanje "ledice" u zeleno

        pokrenut_ventilator = true;

        button4.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }

    private void button2_Click(object sender, EventArgs e)       // gašenje
    ventilatora i postavljanje "ledice" u crveno
    {
        MotorDC.SetMotors(0, 0, 0, 0, 0, 0, 0, 0, 0);

        pokrenut_ventilator = false;

        button4.BackColor = Color.Red;
        timer1boje.Enabled = false;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
        if (pokrenut_ventilator)                                //
        mijenjanje smjera vrtnje ventilatora te provjera ako je već bio upaljen
        {
            if (checkBox1.Checked == true)
            {
                MotorDC.SetMotors(0, 0, 1, 50, 0, 0, 0, 0, 0);
            }

            else
            {
                MotorDC.SetMotors(1, 50, 0, 0, 0, 0, 0, 0, 0);
            }
        }
    }

    private void timer1boje_Tick(object sender, EventArgs e)    // timer za
    blinkanje "ledice" zeleno kad je ventilator upaljen        // ovaj timer
    ne radi stalno već ga pozivamo u gornjim funkcijama
    {
        if (button4.BackColor == Color.Green)
        {
            button4.BackColor = Color.Gray;
        }
        else if (button4.BackColor == Color.Gray)
        {
            button4.BackColor = Color.Green;
        }
    }
}

```

```

private void timer3Zvuk_Tick(object sender, EventArgs e)
{
    if (SenzorZvuka.DohvatiInput() == 0) // hvatamo signal
    sa senzora zvuka te ako je buka prevelika odnosno ako senzor daje 0 pali se dodatni
    ventilator za hlađenje te "ledica"
    {
        MotorStepper.RunMotor1(16000, 10, 1, 0);
        pokrenut_ventilator2 = true;
        button5.BackColor = Color.Green;
        timer1boje.Enabled = true;
    }
    else
    {
        MotorStepper.StopMotor1(0);
        pokrenut_ventilator2 = false;
        button5.BackColor = Color.Red;
        timer1boje.Enabled = false;
    }
}

private void timerGraf_Tick(object sender, EventArgs e) // timer za crtanje
grafa temperature u realnom vremenu
{
    Console.WriteLine("Temperatura:" + temperatura.ToString());

    chartXvalue++;
    chart1.Series[0].Points.AddXY(chartXvalue, temperatura);
    chart1.ChartAreas[0].Axes[0].Maximum = chartXvalue;
    chart1.Refresh(); // osvjezi prikaz grafa
}

private void btnShowData_Click(object sender, EventArgs e)
{
    string file = listStartTime.SelectedValue.ToString();

    chart2.Series[0].Points.Clear(); // obriši postojeće vrijednosti
grafa 2

    // procitaj odabranu datoteku i deserijaliziraj podatke
    FileStream fileFS = new FileStream(file, FileMode.Open, FileAccess.Read);
    // otvaranje novog filestream-a da bi citanje radilo
    StreamReader fileSR = new StreamReader(fileFS);
    string json = fileSR.ReadToEnd();

    List<TempData> Lista = new List<TempData>();

```

```

        Lista = JsonConvert.DeserializeObject<List<TempData>>(json);           //
Deserijalizacija stringa pročitano iz datoteke

        int x = 1;
        foreach(TempData data in Lista)
        {
            chart2.Series[0].Points.AddXY(x, Convert.ToDouble(data.temperatura));
            x++;
        }
        chart2.Refresh();
    }

    // Objekat u koji se spremaju podaci za serijalizaciju i deserijalizaciju
    class TempData
    {
        public string datum { get; set; }
        public string temperatura { get; set; }
    }

    // ListBox item objekat koji nam omogućava da spremimo vrijednost svake stavke
    koja je drugacija od prikazanog teksta
    public class ListBoxItem
    {
        public string Value { get; set; }
        public string Text { get; set; }
    }
}

```


MotorDC.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace SeminarskiRad
{
    class MotorDC
    {
        [DllImport("mtb.dll")]
        public static extern bool InitMotoBee();

        [DllImport("mtb.dll")]
        public static extern bool SetMotors(int on1, int speed1, int on2, int speed2,
int on3, int speed3, int on4, int speed4, int servo);
    }
}
```

MotorStepper.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace SeminarskiRad
{
    class MotorStepper
    {
        [DllImport("stp.dll")]
        public static extern int InitStp();

        [DllImport("stp.dll")]
        public static extern bool RunMotor1(int steps, int interval, int direction,
int outputs);

        [DllImport("stp.dll")]
        public static extern bool StopMotor1(int outputs);
    }
}
```

SenzorZvuka.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace SeminarskiRad
{
    class SenzorZvuka
    {
        [DllImport("mtb.dll")]
        public static extern bool InitMotoBee();

        [DllImport("mtb.dll")]
        public static extern bool Digital_IO(ref int inputs, int outputs);

        public static int DohvatiInput()
        {
            int SenzorZvuka = 0;
            Digital_IO(ref SenzorZvuka, 0);
            return SenzorZvuka;
        }
    }
}
```

WASP.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;

namespace SeminarskiRad
{
    class WASP
    {
        [DllImport("wsp.dll")]
        public static extern bool InitWasp();

        [DllImport("wsp.dll")]
        public static unsafe extern int ReadInputs(int* analog);

        [DllImport("wsp.dll")]
        public static extern void SetOutputs(int bits);

        public static int ProcitajInput() // Citanje inteziteta
        svijetlosti sa senzora
        {
            int[] vrijednosti = new int[4];
            unsafe
            {
                fixed (int* pokazivac = &vrijednosti[0])
            }
        }
    }
}
```

```
        {
            ReadInputs(pokazivac);
        }
    }
    return vrijednosti[0];
}
}
```