Udaljeni nadzor vremenskih uvjeta u prostoriji

Božić, Luka

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:199511

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

UDALJENI NADZOR VREMENSKIH UVJETA U PROSTORIJI

Diplomski rad

Luka Božić

Osijek, 2016. godina

SADRŽAJ

1. UVOD 1

2. KLIJENTKA STRANA 2

2.1. Arduino 2

2.2. Arduino Mega2560 3

2.3. Ethernet*shield* 4

2.4. DHT22 senzor temperature i vlažnosti zraka 5

2.5. DS1307 Real-time Clock modul 6

2.6. LCD ekran 7

3. POSLUŽITELJSKA STRANA 9

3.1. PHP 9

3.2. MySQL 10

4. OBJAŠNJENJE KODA 11

5. ZAKLJUČAK 25

6. LITERATURA 26

POPIS SLIKA 27

ABSTRACT 30

ŽIVOTOPIS 31

PRILOZI 32

1. UVOD

Svakodnevni vremenski uvjeti mogu bitno utjecati na život. Velike promjene ovih uvjeta mogu uzrokovati bitne posljedice ljudima kojima je to bitno. Primjerice, u svakom plasteniku bitno je da su uvjeti pogodni za razvoj bilja, u vinskom podrumu je bitno da je temperatura konstantna, a u skladištima prevelika ili preniska temperatura ili vlaga mogu uništiti robu koja se skladišti. Zbog navedenih, ali i drugih okolnosti često je potreban nekakav sustav nadzora unutarnjih vremenskih uvjeta. Ovakvi sustavi nekad su bili obični analognih termometri, koje čovjek operater mora obilaziti i pregledavati. Napretkom tehnologije pojavili su se i digitalni mjerači sa više funkcija. Unazad desetak godina počeli su razvijati i sustavi koji uvelike operateru olakšavaju posao prikupljajući podatke sa više mjernih mjesta i prikazujući ih na jednoj lokaciji.

Tema ovog diplomskog rada je jedan sustav za udaljeni nadzor vremenskih uvjeta kojeg svaki entuzijast i poznavatelj elektronike može sam sastaviti i postaviti za relativno malo novaca. Prikazat će se neke vrste i načini korištenja *Arudino* platforme za mjerenje vrijednosti, njihov prikaz i spremanje i drugo. *Arduino* platforma te proširenja korištena pri izradi ovog sustava opisana su u drugom poglavlju ovog rada, a u trećem poglavlju opisani su resursi korišteni na poslužiteljskoj strani, točnije tehnologije korištene pri udaljenom spremanju i prikazu prikupljenih podataka. U četvrtom poglavlju detaljno su opisani izvorni kodovi klijenta, *Arduino* sklopa, poslužiteljske strane i internetskog servera.

2. KLIJENTKA STRANA

Na klijentskoj strani nalazi se *Arduino* sklop, mikroupravljačka pločica *Mega2650* s dodatnim priključenim modulima.

2.1. Arduino

Arduino je projekt otvorenog koda (engl. *open source*) koji je napravio brojne alate bazirane na mikroupravljačima od kojih se stvaraju uređaji koji su u mogućnosti upravljati raznim elektroničkim uređajima i prikupljati podatke iz raznih senzora [1]. *Arduino* alati pravljeni su s ciljem da budu jednostavni koliko god je moguće. Postoje mnoge vrste i veličine *Arduino* razvojnih sustava s raznim namjenama i mogućnostima korištenja. Obzirom da je cijeli sustav vrlo jednostavan za korištenje, koriste ga razni tipovi ljudi u razne svrhe, npr. umjetnici, hakeri pa čak i profesionalci za jednostavno dizajniranje, prototipiranje i eksperimentiranje s elektronikom. Projekt je baziran na mikroupravljačkim pločicama raznih proizvođača, a ovisno o pločici različiti su i korišteni mikroupravljači. Mikroupravljači koriste ulaze koji dohvaćaju informacije od korisnika ili iz okoliša i izlaze koji izvršavaju određene radnje u skladu s danim instrukcijama. Ulazi i izlazi mogu biti analogni ili digitalni. Većina nožica mikroupravljača mogu biti korišteni i kao ulazi i kao izlazi (I/O).

Kako bi se na pločice snimio korisnički program prema kojem se Arduino pločice ponašaju koristi se serijsko sučelje, najčešće *USB*. Za samo pisanje upravljačkih programa *Arduino* pruža i vlastito razvojno sučelje (engl. *Integrated development enviroment*) – *Arduino IDE* koje se bazira na programskom jeziku *Processing*, a podržava programske jezike *C* i *C++*. Programski jezik koji *Arduino IDE* primjenjuje gotovo je jednak jeziku *C*, ali pruža i brojne biblioteke za jednostavnije korištenje.

Najviše *Arduino* razvojnih pločica sadrži 8-bitni, 16-bitni ili 32-bitni mikroupravljač tvtrke *Atmel*, ali od 2015. godine počeli su se koristiti i mikroupravljači drugih proizvođača. Najčešće korišteni mikroupravljači su ATmega8, ATmega168, ATmega328, ATmega128 i ATmega256, a većina pločica sadrži i linearni regulator napona od 5Vi 3.3V i kristalni oscilator frekvencije

[1] https://www.arduino.cc/en/guide/introduction, studeni 2016.

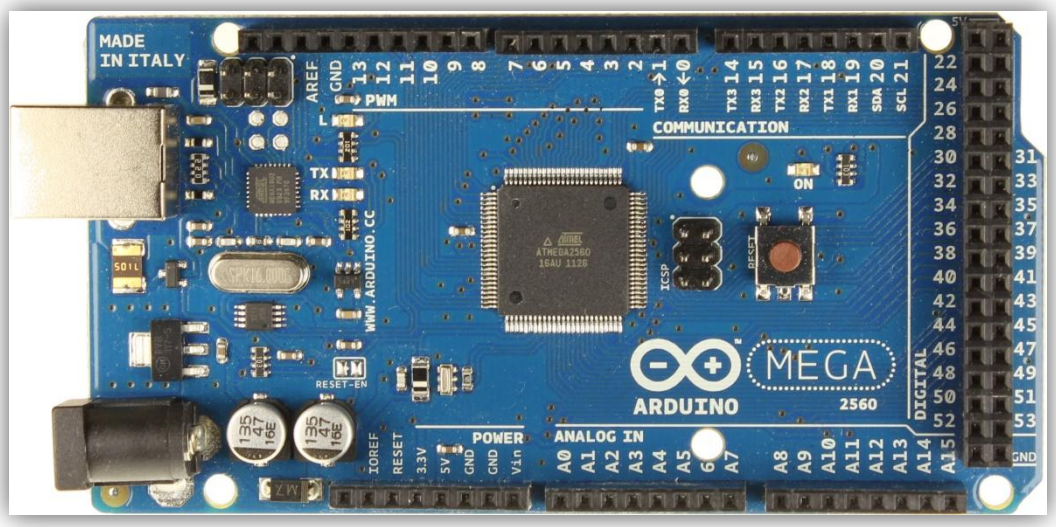
16MHz. U trajnu memoriju mikroupravljača ugrađenih u *Arduino* pločice snimljen je pokretački program (engl. *bootloader*) koji olakšava učitavanje korisničkih upravljačkih programa.

Postoji nekoliko osnovnih službenih *Arduino* pločica, a to su *Tian, Min i, Ethernet, Uno, Leonardo, Yun mini, Yun, Esplora, Micro, Due, ADK, Mega2560* i *Nano,* a najčešće korišteni su *Uno, Mega2560* i *Nano.* Obzirom da je *Arduino* projekt otvorenog koda postoje mnogi proizvođači navedenih pločica i priključaka, od kojih se neki znaju razlikovati od službenih, a pločice je moguće i samostalno izraditi pomoću besplatno dostupnih nacrta.

2.2. Arduino Mega2560

Arduino Mega2560 [2] prikazan na slici 2.1. je razvojna pločica koja se bazira na mikroupravljaču ATmega2560. Uz 54 digitalna ulazno/izlazna priključka (od kojih se 14 može koristiti kao *PWM* izlaz) sadrži i 16 analognih ulaza/izlaza, 4 UART priključka i memoriju od 256KB od kojih 8KB koristi *bootloader,* što znači da je za korisnički program dostupno velikih 248KB. Radna frekvencija je 16MHz, dobivena pomoću kristalnog oscilatora, a maksimalna jakost struje koju pružaju ulazno/izlazni priključci je 40mA. Kako mikroupravljač radi na naponu od 5V ugrađen je regulator napona pa je pločicu moguće spojiti na napon od 6V do 20V, a preporučen napon za spajanje je 7V do 12V istosmjerne struje.

[2] http://world.arduino.org/en/arduino/arduino-mega2560-rev3.html, studeni 2016.



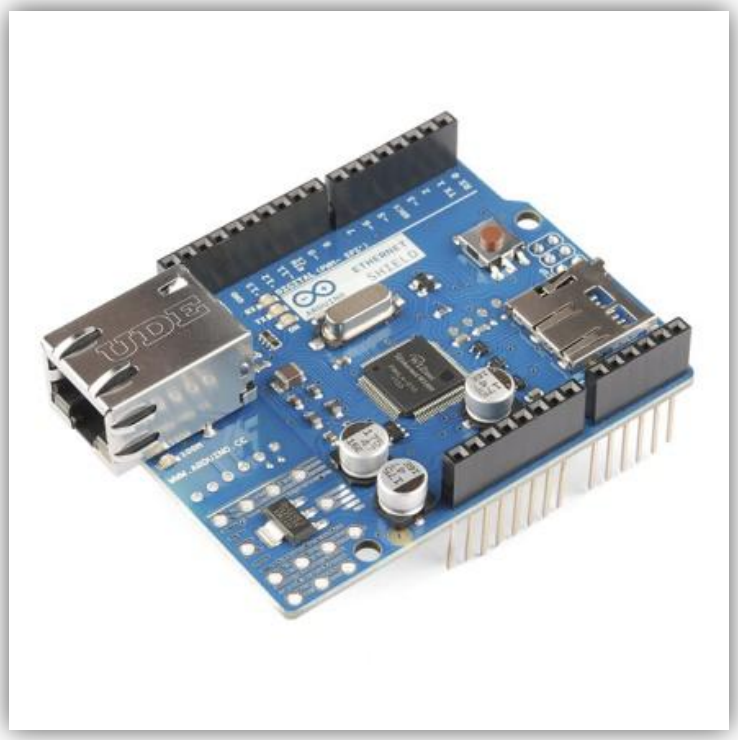
Sl. 2.1. Arduino Mega2560

Uz glavne upravljačke pločice dostupan je i velik broj pločica za proširenja, tzv. *shield*-ova. Ovakve pločice se najčešće samo nataknu na glavnu pločicu ili se povežu kablovima na odgovarajuće priključke, a gotov uvijek ih se više može povezati i koristiti istovremeno. Namjene *shield*-ova su razne, npr. za kontrolu raznih motora, za povezivanje na mrežu i internet, LCD ekran, očitavanje temperature i dr.

2.3. Ethernet*shield*

Arduino EthernetShield [3] prikazan na slici 2.2. omogućava spajanje *Arduino* pločice na ethernet mrežu a time i internet. Ova pločica se jednostavno spoji s osnovnom pločicom, poveže se s mrežom pomocu RJ45 kabla te se uz pravilno postavljene parametra u korisničkom programu vrlo jednostavno povezuje na internet. Mrežni integrirani sklop koji se na ovoj ploči koristi je Wiznet W5500 sa međuspremnikom od 32KB, također radi na 5V, pruža brzinu komunikacije do 100Mb/s, a sa osnovnom *Arduino* pločicom komunicira putem *SPI* priključka.

[3] https://www.arduino.cc/en/Main/ArduinoEthernetShield, studeni 2016.



Sl. 2.2. Arduino Ethernet Shield

Arduino Ethernet Shield na sebi najčešće sadrži i sučelje za *microSD* memorijsku karticu na koju se mogu zapisivati ili s nje čitati razni podaci.

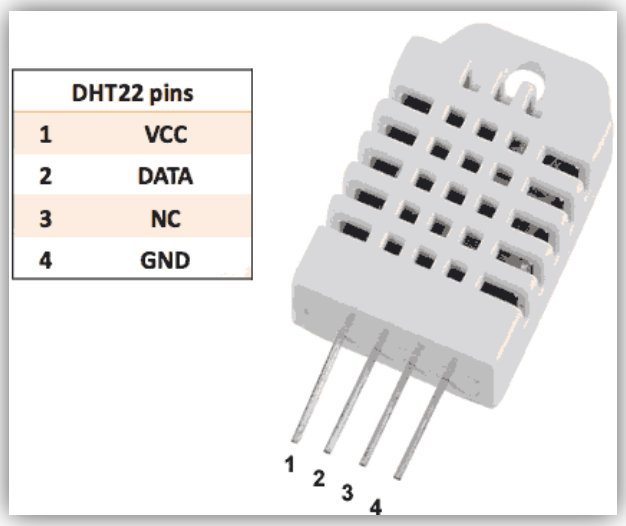
Iako *Ethernet Shield* prekriva osnovnu pločicu na njemu se nalaze priključci s istim rasporedom kao i na osnovnoj pločici pa se funkcionalnost osnovne pločice ne gubi.

2.4. DHT22 senzor temperature i vlažnosti zraka

DHT22 [4] prikazan na slici 2.3. je jeftin, ali pouzdan digitalni senzor temperature i vlažnosti zraka, što znači da ne treba analogni ulaz za komunikaciju sa *Arduino* pločicom. Iako ima 4 nožice za upotrebu su potrebne tri, dvije za napajanje i jedna za komunikaciju. Napon na kojem ovaj senzor radi je 3V do 5V istosmjerne struje, očitava temperature od -40°C do 80°C sa mogućnošću pogreške do 0,5°C i vlažnost od 0 do 100% sa mogućnošću 2% do 5% pogreške.

[4] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf, studeni 2016.

Mana ovog senzora je što je relativno spor, odnosno što mu za očitanje novih vrijednosti treba i do dvije sekunde.



Sl. 2.3. DHT22 senzor temperature i vlažnosti zraka

2.5. DS1307 Real-time Clock modul

Real-time Clock modul (RTC) prikazan na slici 2.4. se koristi kako bi se u sklopovima i uređajima koji nemaju ugrađen sat (kao *Arduino*) mogle koristiti oznake vremena. Ovaj modul baziran je na Maxim DS1307 čipu te ga je moguće spojiti na dva uređaja istovremeno. Za komunikaciju s *Arduino*-om koristi se I2C sučelje. Na modulu je ugrađen priključak za CR2032 bateriju kako bi „zapamtio“ točno vrijeme i datum čak i kad nije napajan od strane *Arduino*-a.

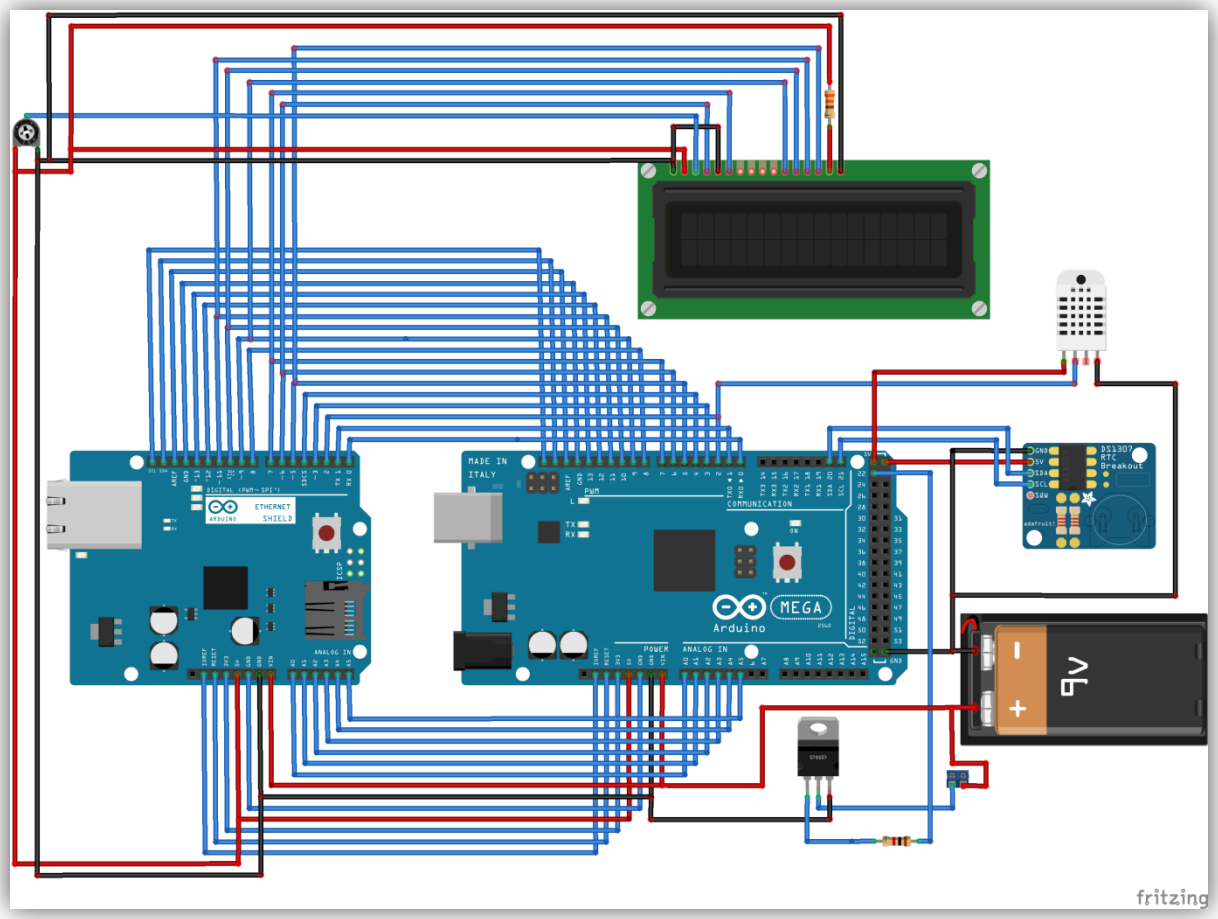
Sl. 2.4. DS1307 Real-time Clock modul

2.6. LCD ekran

LCD (engl. *Liquid crystal display*) prikazan na slici 2.5. ekran koristi se kako bi korisnik u neposrednoj blizini sklopa mogao vidjeti trenutno stanje izvršavanje programa ili za neku drugu namjenu. Ovakvi sklopovi se često koriste u sklopovima baziranim na *Arduino* platformi zbog jednostavnosti i velikih mogućnosti korištenja. U ovom sklopu koristi se ekran koji podržava ispis za ukupno 32 znaka, odnosno po 16 znakova u dva retka. Za komunikaciju sa *Arduino* pločicom koristi se ukupno 6 priključaka, a pomoću priključenog potenciometra moguće je i mijenjati kontrast slova pri ispisu.

Sl. 2.5. LCD ekran

Shema sklopa klijentske strane sustava, *Arduino* sklopa s priključenim modulima prikazana je na Sl. 2.6. Shema klijentske strane sustava



Sl. 2.6. Shema klijentske strane sustava

3. POSLUŽITELJSKA STRANA

Poslužiteljsku stranu čine *PHP* mrežna stranica i *mySQL* baza podataka.

3.1. PHP

PHP je skriptni jezik koji se izvodi na poslužiteljskoj strani. Prvenstveno je stvoren za uporabu u internet programiranju, ali koristi se i kao programski jezik opće namjene. Kreirao ga je 1994.godine Rasmus Lerdorf [5]. U početku je ime *PHP* bilo kratica od „Personal Home Page“, ali danas ima prošireno značenje „PHP: Hypertext Preprocessor“ [6].

PHP dio koda piše se unutar oznaka *„<?php“* i *„?>“*. PHP kod se može kao što vidimo na slici 3.1., ali ne mora kao što vidimo na slici 3.2. ugraditi unutar *HTML* koda stranice pa će tako sljedeća dva odsječka koda dati jednak rezultat:

```
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo '<p>Hello World</p>'; ?>
    </body>
</html>
```

Sl. 3.1. Prvi primjer PHP koda

```
<?php echo 'Hello World'; ?>
```

Sl. 3.2. Drugi primjer PHP koda

[5] http://php.net/manual/en/history.php.php
[6] http://php.net/manual/en/preface.php

PHP se može postaviti na većini poslužitelja, podržavan je od većine operacijskih sustava i platformi, i može se koristiti s mnogim sustavima relacijskih baza podataka. Većina pružatelja web usluga svojim klijentima pruža podršku za *PHP* jer pruža mnogo mogućnosti i može se koristiti besplatno, a *PHP Group* svojim korisnicima daje kompletan izvorni kod kako bi ga si mogli prilagoditi, proširi ili razvijati.

3.2. MySQL

MySQL [7] je sustav za upravljanje relacijskim bazama podataka otvorenog koda. Tvorac i vlasnik *MySQL*–a je bila tvrtka *MySQL AB*, a koju danas posjeduje tvrtka *Oracle Corporation*. Uz besplatnu verziju dostupne su i verzije koje pružaju dodatne mogućnosti, ali se plaćaju.

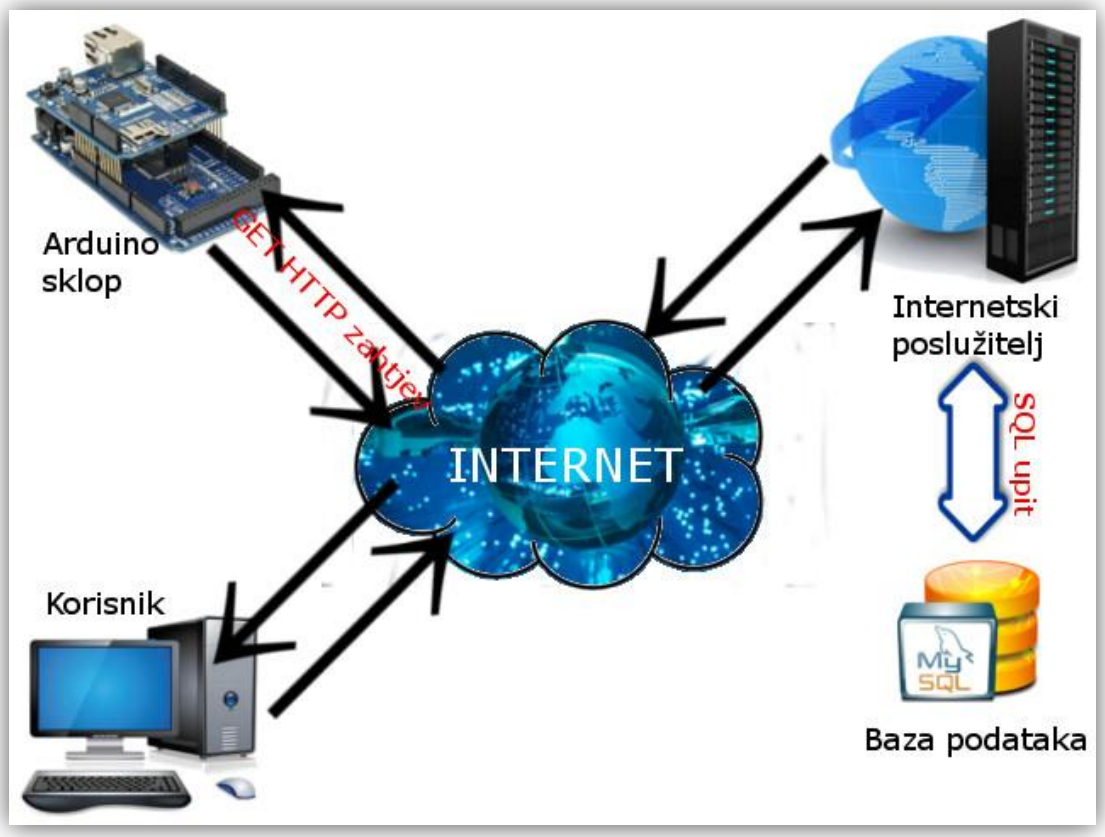
Neke od aplikacija koje koriste *MySQL* baze podataka su i *TYPO3, MODx, Joomla, WordPress, phpBB, MyBB* i *Drupal*. *MySQL* također koristi i velik broj vrlo korištenih internetskih stranica poput *Google*-a, *Facebook*-a, *Twitter*-a, *Flickr*-a i *YouTube*-a. Zbog svojih dobrih performansi, brojne lako dostupne dokumentacije i činjenice da radi na velikom broju operacijskih sustava *MySQL* je najčešće korišten sustav za upravljanje bazama podataka.

Postoje brojna korisnička sučelja kojima je moguće upravljati *MySQL* sustavima. Prosječnim korisnicima najčešći izbor je *phpMyAdmin*. To je besplatan alat otvorenog koda, pisan u *PHP*-u, kojem je namjena upravljanje *MySQL*-om iz internetskog preglednika. Iz ovog sučelja moguće je izvršavati brojne zadatke poput stvaranja, izmjenjivanja ili brisanja baza podataka, tablica, polja i redova, izvršavanja *SQL* upita ili upravljanja korisnicima i dopuštenjima. Ovaj softver dostupan je u 78 jezika [8], a održava ga *The phpMyAdmin Project*.

[7]http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html
[8]https://www.phpmyadmin.net/translations/

4. OBJAŠNJENJE KODA

Kako je već spomenuto, izrađeni sustav sastoji se od dva dijela, *Arduino* sklop - klijent i internetski poslužitelj. Shema sustava prikazana je na Sl. 4.1.



Sl. 4.1. Shema sustava

Izvorni kod sklopa pisan je u *Arduino IDE*, službenom razvojnom okruženju Arduino sustava, a programski jezik je prilagođeni C++. C++ je sekvencijalni kod u kojem je bitan redoslijed pisanja naredbi, jer se naredbe izvršavaju onim redoslijedom kojim su i napisane. Na početku svakog Arduino programa navode se korištene biblioteke kao što vidimo na slici 4.2.

```
#include "DHT.h"
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <TimeLib.h>
#include <Wire.h>
#include <DS1307RTC.h>
```

Sl. 4.2. Poziv korištenih biblioteka

Nakon pozvanih biblioteka definiraju se globalne varijable programa kao što vidimo na slici 4.3.

```
String vrijeme;
int sat = 0;
int minuta = 0;
int sekunda = 0;
String satString;
String minutaString;
String sekundaString;
String datum;
int mjesec = 0;
int dan = 0;
String mjesecString;
String danString;
```

Sl. 4.3. Globalne varijable za vrijeme i datum

Na slici 4.3. prikazane su samo varijable koje se koriste kod formatiranja datuma i vremena, a sve ostale mogu se vidjeti u izvornom kodu u prilogu 1. Svaki Arduino kod mora sadržavati funkcije *voidsetup()* i *voidloop()*. Sljedeći dio koda je dakle funkcija *voidsetup()* u kojoj se podešavaju ulazni i izlazni priključci i moduli spojeni na glavnu Arduino pločicu kao što vidimo na slici 4.4. i 4.5.

```
void setup()
{
  pinMode(10, OUTPUT);
  pinMode(ventilator, OUTPUT);
  digitalWrite(10, HIGH);

  Serial.begin(9600);

  setSyncProvider(RTC.get);
  if (timeStatus() != timeSet)
    Serial.println("Nije moguca sinkronizacija sa RTC-om");
  else
    Serial.println("RTC je postavio vrijeme sustava");
```

Sl. 4.4. Postavljanje priključka 10 kao izlaz, pokretanje serijske komunikacije i RTC-a

```
lcd.begin(16, 2);
lcd.print("Palim sustav...");
dht.begin();
if (!SD.begin(chipSelect)) {
  Serial.println("Inicijalizacija kartice nije uspjela ili nema kartice");
  return;
}
Serial.println("Kartica inicijalizirana");
if (Ethernet.begin(mac) == 0) {
  Serial.println("Nije uspjela konfiguracija etherneta pomocu DHCP-a!");
  Ethernet.begin(mac, ip);
  delay(1000);
  Serial.println("Spajanje...");
}
```

Sl. 4.5. Pokretanje LCD ekrana, senzora vlage i temperature, SD kartice i mrežnog modula

Funkcija *voidsetup()* izvršava se samo jednom, pri pokretanju sustava, a nakon nje slijedi funkcija *voidloop()* čije izvršavanje se ponavlja dok god je sustav upaljen ili ne dođe do nekog zastoja. U prvom dijelu te funkcije očitavaju se vlažnost i temperatura zraka i zapisuju se u varijable *h* i *t* te ispisuju na LCD ekran i serijski monitor za kontrolu toka programa kao što vidimo na slici 4.6.

```
void loop() {
  int h = dht.readHumidity();
  int t = dht.readTemperature();
  if (isnan(t) || isnan(h)) {
    Serial.println("Nije moguce ocitati vrijednosti sa DHT-a");
  }
  else {
    lcd.setCursor(0, 0);
    lcd.print("Vlaznost: "); lcd.print(h);
    Serial.print("\nVlaznost: ");
    Serial.print(h);
    lcd.setCursor(0, 1);
    lcd.print("Temp: "); lcd.print(t);
    Serial.print("\tTemperatura: ");
    Serial.println(t);
  }
```

Sl. 4.6. Očitavanje vlažnosti i temperature i ispis na LCD zaslon i serijski monitor

Nakon očitanih vlažnosti i temperature, u odgovarajuće varijable učitavaju se trenutno vrijeme i datum iz RTC-a te se oblikuju za dvoznamenkasti ispis brojeva kao što vidimo na slici 4.7.

```
if (timeStatus() == timeSet) {
  sat = hour();
  minuta = minute();
  sekunda = second();
  mjesec = month();
  dan = day();
  if (sat >= 0 && sat < 10) {
    satString = String("0" + String(sat));
  }
  else {
    satString = String(sat);
  }
```

Sl. 4.7. Očitavanje datuma i vremena i njihovo oblikovanje

Tako oblikovane vrijednosti datuma, vremena i vremenskih uvjeta spremaju se na priključenu *microSD* memorijsku karticu u oblikovani tekstni dokument *datalog.txt* kao što vidimo na slici 4.8.

```
File dataFile = SD.open("datalog.txt", FILE_WRITE);
if (dataFile) {
  dataFile.print(t);
  dataFile.print("\t");
  dataFile.print(h);
  dataFile.print("\t");
  dataFile.print(vrijeme);
  dataFile.print("\t");
  dataFile.println(datum);
  dataFile.close();
  Serial.println("Uspjesno upisano na karticu");
}
else {
  Serial.println("Pogreska pri otvaranju datalog.txt");
}
```

Sl. 4.8. Zapis podataka na microSD memorijsku karticu

Sljedeća zadaća sklopa je pokretanje ventilatora ukoliko je korisnik to naredio putem web sučelja kao što vidimo na slici 4.9. Funkcijom *charconnect And Read()* putem HTTP zahtjeva *GET* provjerava je li na određenoj stranici i u bazi podataka upisan broj „1“ ili „0“. Ako je rezultat koji funkcija vrati „1“ pali se ventilator, a ako je rezultat „0“ ventilator se gasi. Prethodno stanje se pamti pa ako se dvaput za redom očita ista vrijednost nikakav novi upravljački signal neće biti poslan.

```
char pageValue = connectAndRead();
Serial.println(pageValue);
if (pageValue == '1' && prosli == '0') {
  digitalWrite(ventilator, HIGH);
  prosli = '1';
  Serial.println("Ventilator upaljen");
}
else if (pageValue == '0' && prosli == '1') {
  digitalWrite(ventilator, LOW);
  prosli = '0';
  Serial.println("Ventilator ne radi");
}
else {
  Serial.println("Ne radim nista!");
}
```

Sl. 4.9. Dio koda za upravljanje ventilatorom

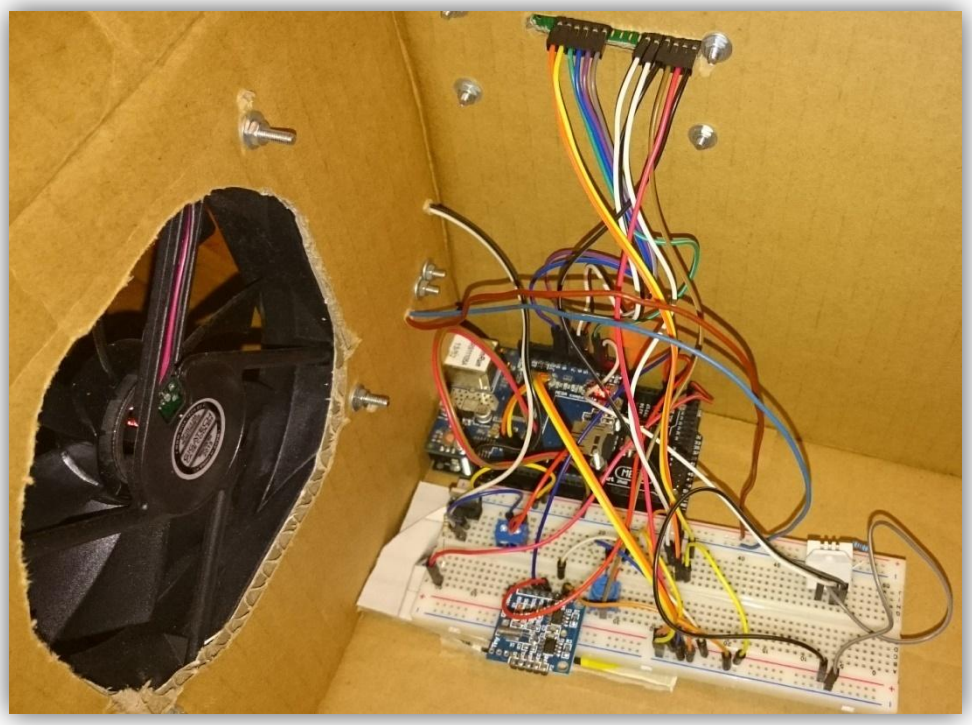
Sljedeća funkcija koja se poziva je *void httpRequest()* koja očitane vrijednosti *GET* HTTP zahtjevom prosljeđuje skripti *add.php*, a koja ih sprema u bazu podataka kao što vidimo na slici 4.10.

```
void httpRequest(int t, int h, String vrijeme, String datum) {
  client.stop();
  if (client.connect(server, 80)) {
    Serial.println("Spojen na posluzitelj");
    Serial.println("Zapisujem ocitane podatke: ");
    Serial.print("Vlaznost: ");
    Serial.print(String(h));
    Serial.print("\tTemperatura: ");
    Serial.println(String(t));
    client.println("GET /pepi/add.php?temp=" + String(t) + "&hum="
    + String(h) + "&time=" + vrijeme + "&date=" + datum + " HTTP/1.1");
    Serial.println("Upisano!");
    client.println("Host: agro-horti.hr");
    client.println("User-Agent: arduino-stanica");
    client.println("Connection: close");
    client.println();
    client.stop();
    lastConnectionTime = millis();
  }
  else {
    Serial.println("Spajanje nije uspjelo");
  }
}
```

Sl. 4.10. Funkcija void httpRequest()

Unutar te funkcije klijent, Arduino, se prvo pokušava povezati s definiranim poslužiteljem. U slučaju uspješnog povezivanja u *GET*zahtjevu prosljeđuje varijable *t* i *h* (očitane temperaturu i vlažnost zraka) te *vrijeme* i *datum*. Ovim zahtjevom pozvana skripta *add.php* primljene podatke sprema u bazu podataka što će biti prikazano u nastavku. Klijent se nakon uspješno odrađenog posla odspaja od poslužitelja, a u slučaju da se nije uspješno uspio povezati s poslužiteljem u serijski monitor upisuje odgovarajuću poruku.

Na Sl. 4.11. Klijentska strana sustava (iznutra)i sliciSl. 4.12. Klijentska strana sustava (izvana) prikazano je kako izgleda cijela sklopljena klijentska strana sustava.



Sl. 4.11. Klijentska strana sustava (iznutra)

Sl. 4.12. Klijentska strana sustava (izvana)

Poslužiteljska strana koda pisana je u *HTML*-u i *PHP*-u. *PHP* skripta koju *Arduino* sklop poziva i pomoću koje se vrši upis očitanih vrijednosti u bazu podataka zove se *add.php* kao što vidimo na slici 4.13. U njoj se nakon ostvarenog spajanja sa *mySQL* bazom podataka dohvaćaju varijable koje su putem *GET* zahtjeva upućene skripti te se *INSERT* upitom upisuju.

```php
<?php
    include("connect.php");

    $link=Connection();

    $temp=$_GET["temp"];
    $hum=$_GET["hum"];
    $time=$_GET["time"];
    $date=$_GET["date"];

    $query = "INSERT INTO ocitanja (`temp`, `hum`, `time`, `date`)
    VALUES ('".$temp."','".$hum."','".$time."','".$date."')";

    mysqli_query($link,$query);
    mysqli_close($link);

    header("Location: index.php");
  exit;
?>
```

Sl. 4.13. Skripta *add.php*

U drugom retku skripte *add.php* poziva se skripta *connect.php* prikazana na slici 4.14. koja služi za povezivanje sa *mySQL* bazom podataka.

```php
<?php

function Connection(){
    $server="localhost";
    $user="username";
    $pass="password";
    $db="database";

    $connection = mysqli_connect($server, $user, $pass, $db);

    if (mysqli_connect_errno())
    {
        echo "Failed to connect to MySQL: " . mysqli_connect_error();
    }
    return $connection;
}

?>
```

Sl. 4.14. Skripta *connect.php*

U ovoj su skripti unutar funkcije *Connection()* navedeni podaci o serveru, korisničkom imenu i lozinki te bazi podataka s kojom se spaja. Pozivom ugrađene funkcije *mysqli_connect* kojoj se predaju navedeni podaci vrši se spajanje na odabranu bazu podataka.

Korisničko sučelje stvoreno je stranicom *index.php*. I u njoj se u početku poziva funkcija za spajanje s bazom podataka. U zaglavlju stranice kao što vidimo na slici 4.15., unutar oznaka *<head>* definirani su osnovni podaci o stranici, naziv, vrsta, jezik te pozvane datoteke koje sadrže informacije o izgledu i uređenju stranice i *JavaScript* skripte koje se koriste za prikaz i funkciju gumba.

```html
<html>

<head>
    <title>Vremenski uvjeti</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" href="css/jquery.mobile-1.4.2.min.css" />
    <link rel="stylesheet" href="css/stil.css" />
    <script src="js/jquery-1.9.1.min.js"></script>
    <script src="js/jquery.mobile-1.4.2.min.js"></script>
    </script>
</head>
```

Sl. 4.15. Zaglavlje stranice *index.php*

Tijelo stranice sadrži dva odjeljaka unutar svojih <*div*> oznaka. U prvom odjeljku sa oznakom *„main“* prikazanom na slici 4.16. dio je koda koji poziva i ispisuje posljednju vrijednost upisanu u bazu podataka i njezinu vremensku oznaku.

```
<div id="main">
    <?php
  while($row = mysqli_fetch_assoc($q)){
      $temp = $row['temp'];
      $hum = $row['hum'];
      $time = $row['time'];
      $date = $row['date'];
      echo "Temperatura: ".$temp."°C <br>";
      echo "Vlažnost: ".$hum."% <br>";
      echo "Vrijeme zapisa: ".$time." <br>";
      echo "Datum zapisa: ".$date." <br>";
    }
    ?>
</div>
```

Sl. 4.16. Odjeljak sa oznakom *"main"*

U prvom dijelu drugog odjeljka napravljena je *JavaScript* funkcija prikazana na slici 4.17. koja obrađuje događaj promjene stanja klikom na gumb. Trenutno gumbom odabrano stanje prosljeđuje se skripti *update.php* pomoću *POST* zahtjeva.

```
<div id="gumb">
    <script type='text/javascript'>
    $(document).ready(function()
    {
        $('input[name=stanje]').change(function()
        {
            $('form').submit();
        });
    });
    function fun_call(arg)
    {
        $.ajax(
        {
            url: 'update.php',
            dataType: 'text',
            type: 'POST',
            data:
            {
                arg: "" + String(arg) + ""
            }
        });
    }
    </script>
```

Sl. 4.17. *JavaScript* funkcija za obradu događaja gumba

U drugom dijelu ovog odjeljka nalazi se dio koda koji služi za prikaz samog gumba kao što vidimo na slici 4.18.

```
<div style="width:150px;display:block;margin:0 auto;" class="containing-element">
    KONTROLA VENTILATORA:
    <select name="flip-min" id="flip-min" onchange="fun_call(this.value);" data-role="slider">
        <option value="0">UPALI</option>
        <option value="1">UGASI</option>
    </select>
</div>
```

Sl. 4.18. Dio koda za prikaz gumba

Kada korisnik klikom na gumb promijeni njegovo stanje novo odabrano stanje se na već navedeni način prosljeđuje skripti *update.php* prikazanu na slici 4.19. Unutar te skripte se obavlja *mySQL* upit koji odabrano stanje, 0 ili 1, zapisuje u bazu podataka.

```php
<?php

include("connect.php");
$link=Connection();

if (isset($_POST['arg']))
{
    $arg = mysqli_real_escape_string($link,trim($_POST['arg']));
}

$update = false;
$update = mysqli_query($link,"UPDATE `ocitanja`
    SET `stanje`='".$arg."' WHERE `id`='1'");
?>
```

Sl. 4.19. Skripta *update.php*

Zadnje upisano stanje se može vidjeti na stranici *ard/indeks.php* prikazanoj na slici 4.20. unutar znakova „<“ i „>“. Na ovu se stranicu spaja i *Arduino* sklop kada obavlja provjeru stanje gumba.

```php
<?php
define('ROOT_PATH', dirname(__DIR__) . '/');
include(ROOT_PATH.'connect.php');

$link=Connection();
$sql_upit="SELECT stanje FROM ocitanja WHERE `id`='1'";

if (!$q=mysqli_query($link, $sql_upit))
    {
        echo "Nismo uspjeli ucitati retke iz baze"."<br>". mysqli_query();
        die();
    }
    if (mysqli_num_rows($q)==0)
    {
        echo "Nema redaka u bazi.";
    }
    else
    {
        $redak=mysqli_fetch_array($q);
        echo "<".$redak["stanje"].">";
    }
?>
```

Sl. 4.20. Stranica *ard/index.php*

Konačan izgled web stranice i korisničkog sučelja može se vidjeti na sljedećoj slici, Sl. 4.21.

Sl. 4.21. Izgled web stranice i korisničkog sučelja

5. ZAKLJUČAK

Vremenski su uvjeti bitni za gotovo svaki oblik života i posao. Biti pravovremeno informiran o njima može biti presudno za uspjeh nekog projekta ili sigurnost ljudi i opreme. Iako postoji i skupa oprema i sustavi za udaljeni nadzor i kontrolu vremenskih uvjeta u ovom radu pokazan je i jednostavan način kako se uz mali novčani trošak može postići zadovoljavajuća razina kontrole. Osim što je ovakav sustav relativno jeftin za napraviti, pruža i mogućnosti za dodatne nadogradnje, poput ugradnje dodatnih senzora ili aktuatora koji bi pridonijeli razini informiranosti i kontrole. Projekti koji koriste sustave otvorenog koda, poput *Arduino*-a često su s popratnom dokumentacijom lako dostupni i na internetu te se lako dolazi do podrške. Uz *Arduino* u ovom radu je korišten i internetski poslužitelj za prikaz i spremanje prikupljenih podataka i kontrolu za što je potrebno znanje internet programiranja. Tehnologije korištene na internetskom serveru, *PHP* i *MySQL*, uz to što su također tehnologije otvorenog koda, pružaju i velike mogućnosti pa se tako i poslužiteljska strana ovog projekta može dodatno proširiti. Takvo proširenje korisniku bi moglo pružiti ljepše korisničko sučelje ili dodatne mogućnosti prikaza podataka.

Sustav udaljenog nadzora vremenskih uvjeta prikazan u ovom radu unatoč svojoj jednostavnosti pruža veliku mogućnost nadzora i kontrole. Zahtjevi za resursima su minimalni, poput male potrošnje električne energije i internetskog prometa, a dijelovi potrebni za izradu sklopa se mogu kupiti prilično jeftino što je u današnje vrijeme jedan od bitnih faktora.

6. LITERATURA

[1] https://www.arduino.cc/en/guide/introduction, studeni 2016.

[2] http://world.arduino.org/en/arduino/arduino-mega2560-rev3.html, studeni 2016.

[3] https://www.arduino.cc/en/Main/ArduinoEthernetShield, studeni 2016.

[4] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf, studeni 2016.

[5] http://php.net/manual/en/history.php.php,studeni 2016.

[6] http://php.net/manual/en/preface.php,studeni 2016.

[7] http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html,studeni 2016.

[8] https://www.phpmyadmin.net/translations/,studeni 2016.

POPIS SLIKA

Sl. 2.1. Arduino Mega2560 4
Sl. 2.2. Arduino Ethernet Shield 5
Sl. 2.3. DHT22 senzor temperature i vlažnosti zraka 6
Sl. 2.4. DS1307 Real-time Clock modul 7
Sl. 2.5. LCD ekran 8
Sl. 2.6. Shema klijentske strane sustava 8
Sl. 3.1. Prvi primjer PHP koda 9
Sl. 3.2. Drugi primjer PHP koda 9
Sl. 4.1. Shema sustava 11
Sl. 4.2. Poziv korištenih biblioteka 12
Sl. 4.3. Globalne varijable za vrijeme i datum 12
Sl. 4.4. Postavljanje priključka 10 kao izlaz, pokretanje serijske komunikacije i RTC-a 13
Sl. 4.5. Pokretanje LCD ekrana, senzora vlage i temperature, SD kartice i mrežnog modula 13
Sl. 4.6. Očitavanje vlažnosti i temperature i ispis na LCD zaslon i serijski monitor 14
Sl. 4.7. Očitavanje datuma i vremena i njihovo oblikovanje 14
Sl. 4.8. Zapis podataka na microSD memorijsku karticu 15
Sl. 4.9. Dio koda za upravljanje ventilatorom 16
Sl. 4.10. Funkcija void httpRequest() 17
Sl. 4.11. Klijentska strana sustava (iznutra) 18
Sl. 4.12. Klijentska strana sustava (izvana) 18
Sl. 4.13. Skripta *add.php* 19
Sl. 4.14. Skripa *connect.php* 20
Sl. 4.15. Zaglavlje stranice *index.php* 20
Sl. 4.16. Odjeljak sa oznakom *"main"* 21
Sl. 4.17. *JavaScript* funkcija za obradu događaja gumba 22
Sl. 4.18. Dio koda za prikaz gumba 22
Sl. 4.19. Skripta *update.php* 23
Sl. 4.20. Stranica *ard/index.php* 23
Sl. 4.21. Izgled web stranice i korisničkog sučelja 24

SAŽETAK

U ovom radu opisana je mogućnost jednostavnog udaljenog nadzora vremenskih uvjeta u nekoj prostoriji. Izrađen je sustav koji se sastoji od klijentske i poslužiteljske strane. Klijentsku stranu tvori *Arudino* sklop sa senzorom koji očitava temperaturu i vlažnost zraka (*DHT22*), *Ethernetshield* za povezivanje s mrežom i internetom, *microSD* memorijska kartica koja služi za lokalni spremanje očitanih vrijednosti, *DS1307 RTC* modulom za praćenje aktualnog vremena i datuma, *LCD* ekranom na kojem se prikazuju zadnje očitane vrijednosti i ventilatorom koji služi za hlađenje. Poslužiteljska strana sastoji se od internetskog poslužitelja na kojemu se nalaze *PHP* skripte koje obrađuju i prikazuju podatke prikupljene *Arduino* sklopom te od *MySQL* baze podataka koja služi za udaljeno trajno spremanje prikupljenih podataka. Korisnik kroz korisničko sučelje u obliku web stranice može pratiti zadnje očitane i zapisane vrijednosti te kontrolirati rad ventilatora.

Ključne riječi: vremenski uvjeti, Arduino, internet, DHT22, PHP, MySQL,

ABSTRACT

Remote weather conditionsurveillance in a room

This paper describes the possibility of remote weather condition surveillance in a room. A system composed of a client and a server is made. A client consists of Arduino platform with the temperature and humidity sensor (DHT22), Ethernet shield used for connecting to network and Internet, a microSD memory card used for local storing of the sensed values, DS1307 RTC module for momentary time and date monitoring, LCD screen that shows the last sensed values, and the ventilator for system cooling. A server consists of Internet server with PHP scripts which process and display data gathered by Arduino platform, and MySQL data base used for remote permanent storing of collected data. The user can monitor the last read and written values using user interface, and he can also control the activity of a ventilator.

Keywords: weather conditions, Arduino, Internet, DHT22, PHP, MySQL

ŽIVOTOPIS

Luka Božić rođen je 08.10.1987. godine u Vinkovcima, Republika Hrvatska. Osnovnu školu „A. G. Matoš“ pohađao je u Vinkovcima. Nakon toga upisuje gimnaziju u Vukovaru. Uspješno položenim prijemnim ispitom na Elektrotehničkom fakultetu u Osijeku 2007.godine postaje redovan student na stručnom studiju. Upisuje razlikovnu godinu te završava sveučilišni preddiplomski studij 2012.godine. Trenutno pohađa diplomski studij na istom fakultetu.

Potpis:

_____

PRILOZI

Prilog A:Programski kod klijentska strana

docu.dox.ino:

```
//pozivanje potrebnih biblioteka
#include "DHT.h"
#include <LiquidCrystal.h>
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <TimeLib.h>
#include <Wire.h>
#include <DS1307RTC.h>

//varijable za vrijeme i datum
String vrijeme;
int sat = 0;
int minuta = 0;
int sekunda = 0;
String satString;
String minutaString;
String sekundaString;

String datum;
int mjesec = 0;
int dan = 0;
```

```
String mjesecString;
String danString;

//varijabla za korištenje SD kartice
const int chipSelect = 4;

//varijable za podešavanje ethernet shield-a
char server[] = "agro-horti.hr"; //adresa servera
String location = "/pepi/ard/index.php HTTP/1.1"; //adresa na serveru i tip dokumenta (http/1.1)
IPAddress ip(192, 168, 1, 177); // IP adresa ethernet shield-a ako DHCP ne dodijeli adresu
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
}; //MAC adresa ethernet shielda
EthernetClient client; // inicijalizacija varijable client koja se korisiti pri komunikaciji ethernet shieldom

char inString[32]; // string za ulazne citanje dolaznih podataka
int stringPos = 0; // brojac indeksa stringa (duljine stringa)
boolean startRead = false; // cita li se string ili ne
unsigned long lastConnectionTime = 0; // posljednje vrijeme kad se spojilo na server (ms)
const unsigned long postingInterval = 10L * 1000L; // vremenski razmak izmedu azuriranja (ms)
// "L" treba za brojeve tipa "long"
char prosli = '0'; //varijabla koja provjerava prethodno stanje ventilatora (ukljucen ili iskljucen)

#define DHTPIN 2     // pin na koji je spojen senzor temperature i vlage
#define DHTTYPE DHT22   // tip senzora temperature i vlage
DHT dht(DHTPIN, DHTTYPE); // inicijalizacija varijable tipa DHT
```

```
LiquidCrystal lcd(5, 6, 7, 8, 9, 3); // inicijalizacija varijable lcd, sa pinovima na koje je spojen lcd ekran


int ventilator = 22;


void setup()
{
  // postavljanje pinova kao ulaz ili izlaz
  pinMode(10, OUTPUT);
  pinMode(ventilator, OUTPUT);
  digitalWrite(10, HIGH);

  Serial.begin(9600); // pokretanje serijske komunikacije

  setSyncProvider(RTC.get);   // funkcija za dohvacanje vremena iz RTC-a
  if (timeStatus() != timeSet)
    Serial.println("Nije moguca sinkronizacija sa RTC-om");
  else
    Serial.println("RTC je postavio vrijeme sustava");

  while (!Serial) {}

  lcd.begin(16, 2); // inicijalizacija LCD-a, 16 stupaca i 2 retka
  lcd.print("Palim sustav..."); // pocetna poruka na LCD-u

  dht.begin(); // inicijalizacija senzora temperature i vlage
  // inicijalizacija SD kartice
```

```
  if (!SD.begin(chipSelect)) { //  provjera je li SD kartica pristuna i moze li ju se inicijalizirati
    Serial.println("Inicijalizacija kartice nije uspjela ili nema kartice");
    return;
  }
  Serial.println("Kartica inicijalizirana");

  // Pokretanje etherneta i spajanje na server
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Nije uspjela konfiguracija etherneta pomocu DHCP-a!");
    Ethernet.begin(mac, ip);    // konfiguriraj pomocu IP-a umjesto DHCP-a
    delay(1000);
    Serial.println("Spajanje...");
  }
  delay(1500);
  lcd.clear(); // obrisi sve sa LCD-a
}

void loop() {
  // za ocitavanje teemperature i vlage treba oko 250 millisekundi
  int h = dht.readHumidity(); // ucitavanje vlage u varijablu h
  int t = dht.readTemperature(); // ucitavanje temperature u varijablu t

  // provjera jesu li dohvacene velicine ispravne, ako nisu broj nesto ne valja (NaN)
  if (isnan(t) || isnan(h)) {
    Serial.println("Nije moguce ocitati vrijednosti sa DHT-a");
  }
  else {
```

```
  lcd.setCursor(0, 0);
  lcd.print("Vlaznost: "); lcd.print(h);
  Serial.print("\nVlaznost: ");
  Serial.print(h);

  lcd.setCursor(0, 1);
  lcd.print("Temp: "); lcd.print(t);
  Serial.print("\tTemperatura: ");
  Serial.println(t);
 }
if (timeStatus() == timeSet) {
  // ucitavanje trenutnog vremena u varijable sat, minuta i sekunda
  sat = hour();
  minuta = minute();
  sekunda = second();

  // ucitavanje trenutnog datuma u varijable mjesec i dan
  mjesec = month();
  dan = day();

  //ispis sata sa dvije znamenke
  if (sat >= 0 && sat < 10) {
    satString = String("0" + String(sat));
  }
  else {
    satString = String(sat);
  }
```

```
//ispis minuta sa dvije znamenke
if (minuta >= 0 && minuta < 10) {
  minutaString = String("0" + String(minuta));
}
else {
  minutaString = String(minuta);
}

//ispis sekundi sa dvije znamenke
if (sekunda >= 0 && sekunda < 10) {
  sekundaString = String("0" + String(sekunda));
}
else {
  sekundaString = String(sekunda);
}

//ispis mjeseca sa dvije znamenke
if (mjesec >= 0 && mjesec < 10) {
  mjesecString = String("0" + String(mjesec));
}
else {
  mjesecString = String(mjesec);
}

//ispis dana sa dvije znamenke
if (dan >= 0 && dan < 10) {
  danString = String("0" + String(dan));
```

```
  }
  else {
    danString = String(dan);
  }

  vrijeme = String(satString + ":" + minutaString + ":" + sekundaString); // postavljanje stringa vrijeme u oblik za ispis
  datum = String(String(year()) + "-" + mjesecString + "-" + danString); // postavljanje stringa datum u oblik za ispis

  Serial.print(datum);
  Serial.print(" ");
  Serial.println(vrijeme);
 }
 else {
  Serial.println("Vrijeme nije postavljeno, molim postavite vrijeme!");
  Serial.println();
 }

 lcd.setCursor(0, 0); // postavljanje pokazivaca LCD-a na prvi red i prvi stupac



 // stvaranje ili otvaranje datoteke na SD kartici za upisivanje podataka
 File dataFile = SD.open("datalog.txt", FILE_WRITE);
 // ako je datoteka dostupa, pisi u nju:
 if (dataFile) {
  dataFile.print(t);
```

```
    dataFile.print("\t");
    dataFile.print(h);
    dataFile.print("\t");
    dataFile.print(vrijeme);
    dataFile.print("\t");
    dataFile.println(datum);
    dataFile.close();
    Serial.println("Uspjesno upisano na karticu");
  }
  // ako se datoteka nije otvorila ispisi upozorenje
  else {
    Serial.println("Pogreska pri otvaranju datalog.txt");
  }

  httpRequest(t, h, vrijeme, datum); //pokretanje funkcije koja upisuje podatke na internetsku bazu podataka

  char pageValue = connectAndRead(); //pozivanje funkcije za spajanje na server i citanje odgovora
  Serial.println(pageValue); // ispisi odgovor na serijski monitor
  if (pageValue == '1' && prosli == '0') {
    digitalWrite(ventilator, HIGH);
    prosli = '1';
    Serial.println("Ventilator upaljen");
  }
  else if (pageValue == '0' && prosli == '1') {
    digitalWrite(ventilator, LOW);
    prosli = '0';
```

```
    Serial.println("Ventilator ne radi");
  }
  else {
    Serial.println("Ne radim nista!");
  }


  delay(5000); //cekaj 5 sekundi prije ponovnog spajanja
}

char connectAndRead() {
  Serial.println("Spajanje na posluzitelj"); // spajanje sa posluziteljem
  if (client.connect(server, 80)) { // port 80 je standardni kod HTTP posluzitelja
    Serial.println("Spojeno");
    client.print("GET ");
    client.println(location);
    client.println("Host: agro-horti.hr");
    client.println("Connection: close");
    client.println();

    return readPage(); // procitaj odgovor sa posluzitelja
  }
  else {
    return 4;
  }
}
```

```
char readPage() {
  char rtnVal = 0;
  char startRead = 0;

  // connectLoop vrsi kontrolu u slucaju hardverske greske
  int connectLoop = 0;

  while (client.connected())
  {
    while (client.available())
    {
      char ch = client.read();
      Serial.write(ch);
      if (ch == '<') startRead = 1;
      else if (ch == '>') startRead = 0;
      else if (startRead == 1) rtnVal = ch;

      connectLoop = 0; // connectLoop se postavlja u 0 ako stigne paket
    }
    connectLoop++;
    if (connectLoop > 10000) // ako prode vise od 10000 millisekundi od zadnjeg paketa
    {
      Serial.println();
      Serial.println(F("Timeout"));
      client.stop(); // zavrsi vezu sa posluziteljem
    }
    delay(1);
```

```
  }
  Serial.println();
  Serial.println(prosli);
  Serial.println(F("Odspajanje sa servera"));
  client.stop(); // zavrsi vezu sa posluziteljem

  return rtnVal;
}

void httpRequest(int t, int h, String vrijeme, String datum) {
  // zatvori sve veze prije novog spajanja
  // ovo oslobada socket na ethernet shieldu
  client.stop();
  if (client.connect(server, 80)) { // spajanje na posluzitelj i provjera uspjesnog spajanja
    Serial.println("Spojen na posluzitelj");
    Serial.println("Zapisujem ocitane podatke: ");
    Serial.print("Vlaznost: ");
    Serial.print(String(h));
    Serial.print("\tTemperatura: ");
    Serial.println(String(t));
    // slanje GET zahtjeva sa ocitanim vrijednostima kao argumentima
    client.println("GET /pepi/add.php?temp=" + String(t) + "&hum=" + String(h) + "&time=" +
vrijeme + "&date=" + datum + " HTTP/1.1");
    Serial.println("Upisano!");
    client.println("Host: agro-horti.hr");
    client.println("User-Agent: arduino-stanica");
    client.println("Connection: close");
```

```
    client.println();
    client.stop(); // zavrsi vezu sa posluziteljem

    lastConnectionTime = millis(); // zapisi vrijeme kad je ostvareno spajanje
  }
  else {
    Serial.println("Spajanje nije uspjelo"); // ako se nije mogla ostvariti veza
  }
}
```

Prilog B: Programski kodposlužiteljska strana

index.php:

```
<?php
include("connect.php");
$link=Connection();

$q=mysqli_query($link,"SELECT * FROM ocitanja ORDER BY id DESC LIMIT 1");
?>

<html>
<head>
<title>Vremenski uvjeti</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" href="css/jquery.mobile-1.4.2.min.css" />
<link rel="stylesheet" href="css/stil.css" />
<script src="js/jquery-1.9.1.min.js"></script>
```

```
<script src="js/jquery.mobile-1.4.2.min.js"></script>
</script>
</head>
<body>
<div id="main" >
<?php
    while($row = mysqli_fetch_assoc($q)){
      $temp = $row['temp'];
      $hum = $row['hum'];
      $time = $row['time'];
      $date = $row['date'];
      echo "Temperatura: ".$temp."°C <br>";
      echo "Vlažnost: ".$hum."% <br>";
      echo "Vrijeme zapisa: ".$time." <br>";
      echo "Datum zapisa: ".$date." <br>";
    }
 ?>

        </div>
<div id="gumb">
<script type='text/javascript'>
      $(document).ready(function() {
        $('input[name=stanje]').change(function(){
            $('form').submit();
        });
       });
      function fun_call (arg) {
```

```
        $.ajax({

                url: 'update.php',
                dataType: 'text',
                type: 'POST',
                data: { arg: ""+String(arg)+"" }

        });
        }

</script>
<br>
<br>
<div style="width:150px;display:block;margin:0 auto;" class="containing-element">
        KONTROLA VENTILATORA:
<select name="flip-min" id="flip-min" onchange="fun_call(this.value);" data-role="slider">
<option value="0">UPALI</option>
<option value="1">UGASI</option>
</select>
</div>
<br><br>
</div>

        </div>
</body>
</html>
```

add.php:

```
<?php
	include("connect.php");

	$link=Connection();

	$temp=$_GET["temp"];
	$hum=$_GET["hum"];
	$time=$_GET["time"];
	$date=$_GET["date"];

$query = "INSERT INTO ocitanja (`temp`, `hum`, `time`, `date`)
VALUES ('".$temp."','".$hum."','".$time."','".$date."')";

  mysqli_query($link,$query);
	mysqli_close($link);
  header("Location: index.php");
 exit;
?>
```

index.php(arduino):

```
<?php
define('ROOT_PATH', dirname(__DIR__) . '/');
include(ROOT_PATH.'connect.php');

$link=Connection();
```

```
$sql_upit="SELECT stanje FROM ocitanja WHERE `id`='1'";
if (!$q=mysqli_query($link, $sql_upit))
{
echo "Nismo uspjeli ucitati retke iz baze"."<br>". mysqli_query();
die();
}
if (mysqli_num_rows($q)==0)
{
echo "Nema redaka u bazi.";
}
else {
$redak=mysqli_fetch_array($q);
echo "<".$redak["stanje"].">";
}
?>
```

update.php:

```
<?php
include("connect.php");
$link=Connection();


if (isset($_POST['arg'])) {

        $arg = mysqli_real_escape_string($link,trim($_POST['arg']));
```

```
}

$update = false;

$update = mysqli_query($link,"UPDATE `ocitanja` SET `stanje`='".$arg."' WHERE `id`='1'");

?>
```

connect.php:

```
<?php

        function Connection(){
                $server="localhost";
                $user="agrohort_kralj";
                $pass="123456789";
                $db="agrohort_pepi";

                $connection = mysqli_connect($server, $user, $pass, $db);

                if (mysqli_connect_errno())
                        {
                                echo "Failed to connect to MySQL: " . mysqli_connect_error();
 }
                return $connection;
        }
?>
```