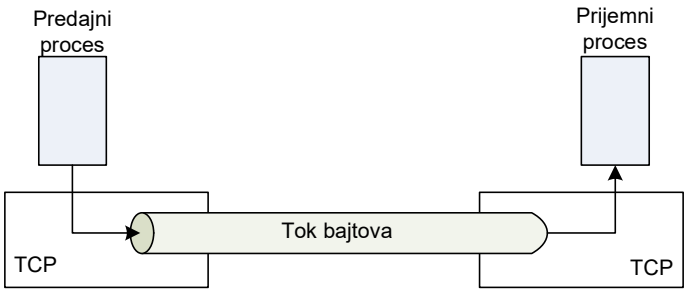
2 TCP i HTTP

2.1 TCP

TCP (*Transmission Control Protocol - Protokol za kontrolu prenosa*) je proces-proces protokol koji, kao i UDP, koristi brojeve portova. Međutim, za razliku od UDP-a, TCP je konekcioni protokol, koji kreira virtuelnu konekciju radi pouzdanog dvosmernog prenosa podataka između dva udaljena procesa. TCP kreira okruženje u kojem izgleda kao da su dva procesa spojena nekom imaginarnom ˝cevi˝ kroz koju teku njihovi podaci (Sl. 1). Predajni proces generiše (upisuje) tok bajtova, a prijemni proces konzumira (čita) bajtove iz ˝cevi˝. Kad se jednom uspostavi konekcija (´cev´) između dva procesa, podaci mogu da teku oba smera. Kad više nema potrebe za komunikacijom, konekcija se zatvara. TCP, za razliku od UDP, je pouzdan protokol. To znači da TCP ispravlja greške u prenosu koje su posledica gubitaka podataka u mreži, pristizanja podataka na odredište izvan redosleda po kome su poslati, dupliciranja poslatih podataka i slično.



Sl. 1 TCP konekcija. Napomena: slika prikazuje prenos podataka u jednom smeru. Identična „cev“ postoji i za prenos podataka u suprotnom smeru

U postupku uspostavljanja TCP konekcije, proces može da igra jednu od dve uloge: klijent ili server. Klijent je proces koji inicira uspostavljanje konekcije, tako što kontaktira odredišni host na broju porta serverskog procesa. S druge strane, serverski proces osluškuje svoj port i čeka na zahtev za uspostavljanje konekcije. Nakon što je konekcija uspostavljena, razlike između klijenta i servera se gube, u smislu da više ne postoje ograničenja u vezi toga koji proces i kada može da šalje svoje podatke. Sve što jedan proces pošalje drugom, taj drugi prima i obrnuto. Konekciju može da raskine bilo koji od dva procesa.

U listingu L. 1 je dat program koji ilustruje upotrebu TCP protokola. Program upravlja jednom LED i senzorom vlažnosti i temperature. U programu je realizovan TCP server. Nakon što se uspostavi TCP konekcija, klijentu se šalje podatak o temperaturi i vlažnosti na svakih $5\ s$. Takođe, klijent može da upravlja LED-om, slanjem kratkih tekstualnih komandi. Komanda „ON“ pali, a komanda „OFF“ gasi LED.

Više detalja o načinu povezivanja senzora vlažnosti i temperature (DTH11) i modula ESP32-S3 dato je u sekciji Zadatak, na kraju ovog dokumenta.

Za programsku realizaciju TCP servera koristi se klasa *WiFiServer* (linija 19). „server“ je objekat ove klase inicijalizovan brojem porta 2589. TCP konekcija je dostupna preko objekta *client* iz klase *WiFiClient* (linija 20). TCP server se pokreće naredbom *begin*() u funkciji *setup*() (linija 37): kaže se da je server otvorio svoj port za slušanje. Nakon otvaranja porta, server je dostupan za konektovanje.

L. 1. Program *TCPServer*

```
1   #include <WiFi.h>
2   #include "DHT11.h"
3
4   const int LED_PIN = 21;           //LED pin
5   const int THM_PIN = 2;            //pin TH modula
6   //mrezna podesavanja:
7   char ssid[] = "***";              //ime WiFi mreze
8   char password[] = "***";          //WiFi lozinka
9   const unsigned int SERVER_PORT = 2589; //broj porta TCP servera
10  //FSM:
11  enum FsmStateType {DISCONNECTED, CONNECTED};
12  FsmStateType fsmState = DISCONNECTED;
13  //TH:
14  long currentTime = 0;       //tekuce vreme
15  long lastTHReadingTime = 0; //vreme poslednjeg ocitavanja TH modula
16  const long TH_READING_PERIOD = 5000; //period ocitavanja TH modula
17  DHT11 dht11(THM_PIN);
18  //TCP:
19  WiFiServer server(SERVER_PORT); //instanciranje TCP servera
20  WiFiClient client;              //instanciranje TCP klijenta
21  char buffer[256];               //prijemni bafer
22  int k = 0;                      //indeks u prijemnom baferu
23
24  void setup() {
25    Serial.begin(115200);
26    pinMode(LED_PIN, OUTPUT);
27    //konektovanje na WiFi mrezu
28    WiFi.begin(ssid, password);
29    while (WiFi.status() != WL_CONNECTED){
30       delay(300);
31    }
32    while (WiFi.localIP() == INADDR_NONE) {
33       delay(300);
34    }
35    //startovanje TCP servera:
36    Serial.println("\nConnected to the WiF network");
37    server.begin();
38    Serial.println("TCP started!");
39  }
40
41  void loop() {
42    currentTime = millis();
43    switch(fsmState){
44      case DISCONNECTED:
45        //provera da li TCP server konektovan
46        client = server.available();
47        if(client){
48          //resetovanje (brisanje) prijemnog bafera:
49          k = 0;
50          buffer[0] = 0;
```

```
51          Serial.println("TCP connection established");
52          fsmState = CONNECTED;
53        }
54        break;
55      case CONNECTED:
56        if(client.connected()){ //sve dok postoji konekcija:
57          if(client.available() > 0){
58            char c = client.read();
59            if(c != '\r'){
60              buffer[k++] = c;
61            } else{
62              buffer[k] = 0; //0 je znak za kraj stringa!
63              Serial.println(buffer);
64              //obrada primljene komande
65              if(!strcmp(buffer,"ON")){
66                digitalWrite(LED_PIN,HIGH);
67              } else if(!strcmp(buffer,"OFF")){
68                digitalWrite(LED_PIN,LOW);
69              }
70              k = 0;
71              buffer[0] = 0;
72              client.flush();
73            }
74          }
75          if(currentTime - lastTHReadingTime > TH_READING_PERIOD){
76            //vreme je za slanje poruke o temperaturi i vlaznosti
77            client.print("T=");client.print(dht11.readTemperature());
78            client.print(" H=");client.println(dht11.readHumidity());
79            lastTHReadingTime = currentTime;
80          }
81        } else{
82          //konekcija je prekinuta
83          Serial.println("TCP connection closed");
84          fsmState = DISCONNECTED;
85        }
86        break;
87    }
88  }
89
```

Program u funkciji *loop*() je realizovan u vidu FSM-a. Stanje FSM je zapravo stanje konekcije. Sve dok konekcija nije uspostavljena, FSM je u stanju DISCONNECTED. Informacija o tome da li je konekcija uspostavljena dobija se preko funkcije *server.available*() u liniji 46. Ukoliko jeste, ova funkcija vraća referencu na objekat tipa *WiFiClient*; ukoliko nije, vraća *null* pointer. Ukoliko je klijent kreiran (*client != null*), FSM prelazi u stanje CONNECTED, uz resetovanje bafera za prijemne podatke (linije 49 - 50). FSM ostaje u stanju CONNECTED sve dok postoji konekcija, što se proverava funkcijom *client.connected*() u liniji 56. Kada konekcija više ne postoji, FSM se vraća u stanje DISCONNECTED (linija 84). Za vreme trajanja konekcije, neprekidno se obavljaju dve aktivnosti: obrada primljenih komandi (linije 57 - 72) i periodično slanje podataka o temperaturi i vlažnosti (linije 75 - 79).

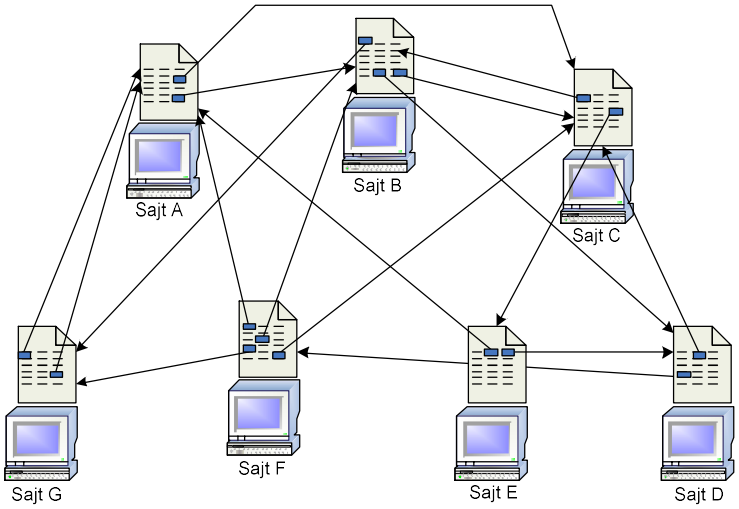
Komanda je oblika jedne linije teksta – string završen znakom na novi red („/r“). Dostupnost primljenih podataka se proverava funkcijom *client.avalilable*() u liniji 57 koja vraća broj primljenih, ali još uvek nepreuzetih bajtova. Ako je ovaj broj veći od nule, u liniji 58 se čita jedan karakter funkcijom *client.read*(). Ako preuzeti karakter nije znak za novi red, ovaj karakter se smešta u prijemni bafer (linija 60). Ako jeste znak za novi red, u bafer se upisuje 0 (znak za kraj stringa) i prelazi se na obradu komande. Postoje samo dve komande. Ako primljena linija teksta glasi „ON“, LED se pali (linija 66); ako glasi „OFF“

LED se gasi (linija 68). Kao što znamo iz C-a, funkcija *strcmp*() poredi dva stringa i vraća 0 u slučaju jednakosti. Ako primljena linija nije ni ON ni OFF, komanda nema nikakvog efekta. Nakon obrade komande, prijemni bafer se prazni (linija 70 - 72). Takođe, odbacuju se bilo kakvi podaci koji se eventualno pristigli posle znaka za novi red, a nisu preuzeti. To se postiže funkcijom *client.flush*() u liniji 72. Na ovaj način, pripremili smo se za prijem sledeće komande.

U linijama 77 i 78 vidimo kako se šalju podaci klijentu. Funkcije *print*() i *println*() imaju isto dejstvo kao i prilikom štampanja na serijski monitor, samo što su sada podaci usmereni ka TCP konekciji.

2.2 Veb (HTTP)

World Wide Web (ili samo *Web*, tj. veb) je kolekcija ogromnog broja elektronskih dokumenata sačinjenih od povezanih *veb stranica* (Sl. 2). Stranice su raspoložive korisnicima veba u vidu datoteka smeštenih na milionima računara distribuiranih po Internetu. Svaka stranica može sadržati pokazivače (tzv. *linkove* ili *hiperveze*) na druge stranice koje mogu biti smeštene na istom ili na nekom drugom računaru (ili, u terminologiji veba, *veb sajtu*). Klikom na hipervezu, korisnik prelazi na sledeću stranicu, a ovaj proces se može nastaviti u nedogled. Koncept stranica koja sadrži pokazivače na druge srodne stranice naziva se *hipertekstom*. Stranica može sadržati i delove koji nisu tekstualni (npr. slike).



Sl. 2 Veb – distribuirana kolekcija povezanih dokumenata

Da bi se veb implementirao na Internetu koriste se dve glavne komponente: veb pregledač i veb server. *Veb pregledač* (*Web browser*) je aplikacioni program koji korisnik koristi da bi pristupio stranici i prikazao je. Najpoznatiji veb pregledači su *Google Chrome*, *Microsoft Edge* i *Mozilla Firefox*. Veb pregledač igra ulogu klijenta koji stupa u vezu sa odgovarajućim *veb serverom* da bi dobio primerak tražene veb stranice. Pregledač pribavlja traženu stranicu, interpretira komande za formatiranje teksta i tekst prikazuje na ekranu monitora. Delovi teksta koji sadrže hiperveze na druge stranice su obično naglašeni (podvučeni ili prikazani u odgovarajućoj boji). Klikom na naglašeni tekst, pregledaču se izdaje instrukcija da pribavi i prikaže stranicu na koju ukazuje izabrana hiperveza. Pregledač pribavlja stranicu bez ikakve pomoći korisnika, a nova stanica može biti uzeta sa istog servera sa kojeg je pribavljena i prva ili sa nekog drugog servera.

URL

URL (*Uniform Resource Locator*, ili univerzalni lokator resursa) je putanja do određenog sadržaja na Internetu. Često se naziva i veb adresa. Sadržaj na koji upućuje URL može biti veb stranica, slika ili bilo koja datoteka koja se nalazi na određenom serveru. Primer tipičnog URL-a:

http://www.abcd.com/products.htm

URL se sastoji iz tri dela: *a*) protokol koji se koristi za pribavljanje veb stranice (deo *http*); *b*) adresa servera na kome je stranica locirana, a koja može biti zadata bilo u obliku simboličkog imena bilo u obliku IP adrese (deo *www.abcd.com*) i *c*) lokalno ime koje na jedinstveni način ukazuje na konkretnu stranicu - najčešće ime fajla na hard disku u kome je stranica zapamćena (deo *products.htm*). Deo za ime fajla može biti izostavljen. U tom slučaju, zahtev se preusmerava na glavnu (podrazumevanu) stranicu veb servera.

HTTP

Protokol HTTP je osnovni protokola veba. To je klijent-server protokol aplikacionog nivoa, koji se koristi za komunikaciju između veb klijenta (pregledača) i veb servera. U većini slučajeva, veb klijent zahteva veb stranicu, a veb server isporučuje njenu kopiju. Za transport poruka, HTTP koristi protokol TCP, podrazumevano na portu 80. Veb server neprekidno osluškuje TCP port 80, čekajući da neki pregledač zatraži otvaranje TCP konekcije. S druge strane, pregledač koji želi da pribavi veb stranicu, inicira otvaranje TCP konekcije na portu 80 sa serverom koji čuva stranicu. Kada je TCP konekcija uspostavljena, pregledač šalje serveru poruku zahteva koja sadrži putanju i ime traženog fajla. U poruci odgovora, veb server šalje pregledaču traženi fajl i po završenom prenosu zatvara TCP konekciju.

Sl. 3 Format HTTP poruke: (a) format zahteva; (b) format odgovora

HTTP zahtev

Kada se na veb stanici koja je prikazana u pregledaču klikne na neki link, pregledač analizira URL na koji ukazuje link (URL je obično skriven iza teksta linka) i izdvaja iz URL-a adresu servera i ime fajla. Zatim, pregledač uspostavlja TCP konekciju sa serverom na portu 80, i kroz otvorenu konekciju šalje HTTP zahtev koji sadrži ime fajla.

HTTP zahtev je tekst koji se sastoji od određenog broja linija (Sl. 3(a)). Svaka linija je završena kombinacijom specijalnih karaktera $0x0D$, $0x0A$ (CR LF), koji označavaju prelazak u novi red. Format HTTP zahtev je prikazan na Sl. 3(a). Deo „Metod“ sadrži jednu liniju teksta oblika:

`tip_zahteva ime_fajla verzija_HTTP_protokola`

Za pribavljanje veb stranica koristi se tip zahteva GET. Aktuelna verzija HTTP protokola je 1.1. S obzirom na to, prva linija HTTP zahteva je oblika:

```
GET ime_fajla HTTP/1.1
```

Deo „Zaglavlje“ sadrži određeni broj linija od kojih svaka nosi neku specifičnu informaciju koja bi mogla biti od interesa veb serveru, npr: IP adresa klijenta, tip pregledača iz kojeg je zahtev upućen, koje vrste sadržaja pregledač može da protumači i slično. Nakon zaglavlja sledi prazna linija, a onda opcioni deo za dodatni sadržaj zahteva. Za zahteve tipa GET, ovaj dodatni deo ne postoji. Ispod je dat sadržaj HTTP poruke zahteva koju je pregledač (konkretno *Chrome*) uputio serveru *www.abced.com* tražeći od njega stranicu *products.htm*.

```
GET /products.htm HTTP/1.1
Host: 192.168.1.3
Connection: keep-alive
Cache-Control: max-age=0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q
=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,sr;q=0.6,hr;q=0.4
```

HTTP odziv

Poruka HTTP odziva je sličnog formata (Sl. 3(b)). Prva linija sadrži status odziva. Ukoliko je veb server prihvatio zahtev i pronašao traženi fajl, statusna linija je oblika:

```
HTTP/1.1 200 OK
```

U delu „Zaglavlje“ nalazi se nekoliko linija od kojih svaka sadrži neku konkretnu informaciju o stranici koja će biti poslata ili neku instrukciju veb klijentu. Ilustracije radi, ispod su date tri tipične linije zaglavlja:

```
Content-type: text/html
Connection: close
Refresh: 5
```

Content-type definiše tip datoteke koju veb server vraća veb klijentu. Vrednost ovog zaglavlja, napisana nakon dvotačke, znači da se u odzivu vraća tekstualni dokument formatiran HTML-om. Linija zaglavlja „*Connection*: *close*“ nosi obaveštenje klijentu da će server da zatvori TCP konekciju nakon što pošalje kompletan sadržaj stranice. Linija zaglavlja „*Refresh:5*“ predstavlja instrukciju klijentu da ponovo zatraži istu stranicu nakon 5 s.

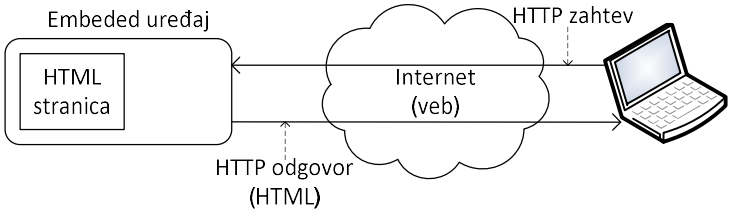
Nakon zaglavlja sledi jedna prazna linija, a onda i sadržaj tražene datoteke. U najvećem broju slučajeva datoteka sadrži veb stranicu u obliku tekstualnog dokumenta formatiranog u jeziku HTML.

HTTP u embeded sistemima

HTTP je ključni protokol za komunikaciju putem Interneta, ali se koristi i u embeded sistemima za realizaciju veb interfejsa i omogućavanje daljinskog pristupa i upravljanja. Veb interfejs pojednostavljuje interakciju sa embeded uređajem, tako što omogućava korisniku da pristupi uređaju putem veb pregledača na računaru, pametnom telefonu, tabletu i sl., i to radi praćenja trenutnog stanja uređaja, očitavanja rezultata merenja, pa čak i izvršavanja osnovnih upravljačkih funkcija.

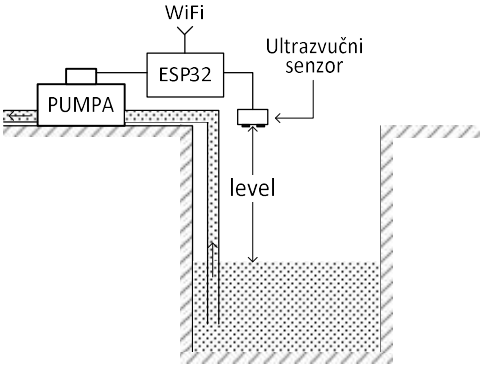
HTTP se relativno jednostavno integriše sa aplikacionim softverom na embeded uređaju. Osnovni princip prikazan je na Sl. 4: embeded uređaj implementira HTTP server kome je dodeljen odgovarajući URL, i u svojoj memoriji čuva veb stranicu u HTML formatu. Kada korisnik putem pregledača pošalje HTTP zahtev, uređaj kao odgovor vraća veb stranicu sa svim relevantnim podacima, koja se zatim

prikazuje u pregledaču. Korisnik, ukoliko je to predviđeno, može nastaviti interakciju sa uređajem izborom opcija na samoj stranici. Način na koji se ova funkcionalnost ostvaruje biće detaljno objašnjen u nastavku.



Sl. 4 Primena HTTP-a u embeded sistemima

Kao primer primene, na Sl. 5 je prikazan sistem za navodnjavanje, koji se sastoji iz ultrazvučnog senzora za merenje nivoa vode u bunaru, pumpe i ESP32-S3 modula, koji implemetira veb interfejs. Putem veb interfejsa, udaljeni korisnik može da dobije uvid u trenutni nivo vode u bunaru i, po potrebi, uključi ili isključi pumpu.



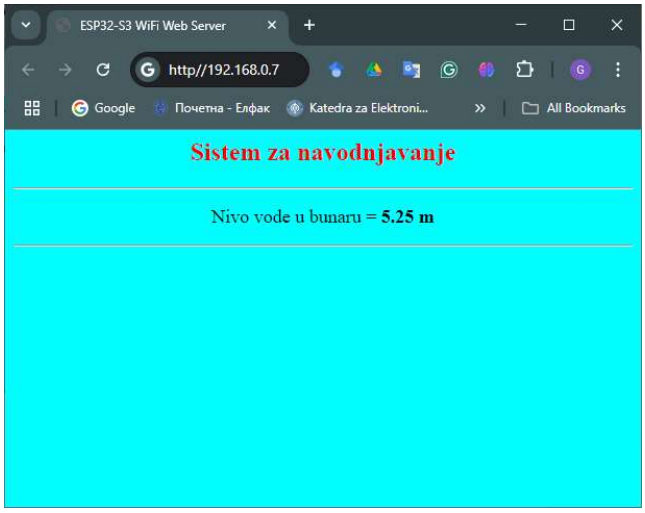
Sl. 5 Sistem za navodnjavanje sa veb interfejsom

HTML

Veb stranice se pišu u jeziku HTML (*HyperText Markup Language* – hipertekstualni markerski jezik). Pomoću HTML-a mogu se kreirati veb stranice koje sadrže tekst, grafiku i hiperveze ka drugim veb stranicama. Pojam *markup language* (markerski jezik) znači da HTML služi za opis načina na koji su dokumenti formatirani, odnosno kako će biti prikazani u veb pregledaču. HTML definiše skup komandi za formatiranje koje, ugrađene u prvobitni tekst dokumenta, daju uputstva za prikazivanje sadržaja dokumenta. Ove komande se nazivaju tagovima. Tagovi se pišu između znakova < i >, a većina tagova ide u parovima. Na primer, tag označava početak, tag kraj boldiranog teksta. Pojedini tagovi nemaju svoj par. Na primer, tag <br> označava prelazak na novi red, a tag <hr> iscrtava horizontalnu liniju duže cele širine stranice. Na Sl. 7 je prikazana jedna jednostavna veb stranica, koja prikazuje izmereni nivo vode u bunaru. Na istoj slici je dat i HTML kôd koji se krije iza ove stranice.

Celokupni sadržaj stranice uokviren je tagovima <html> i </html>. U okviru stranice postoje dva segmenta: zaglavlje, uokvireno tagovima <head> i </head>, i telo stranice, uokvireno tagovima <body> i </body>. U zaglavlju se piše naslov stranice, omeđen tagovima <title> i </title>. Naslov stranice se ne ispisuje na samoj stranici, već u naslovnoj traci pregledača. Druga linija u zaglavlju stranice (**`<link rel="icon" href="data:,">`**) isključuje podrazumevanu opciju pregledača da automatski pribavlja tzv. "*favorite icon*", tj. sličicu malih dimenzija koja se koristi za vizuelnu identifikaciju veb sajta u tabu i adresnoj traci pregledača.

Sadržaj stranice je smešten u telu stranice. Pojedinim tagovima se mogu pridružiti atributi. Na primer, u tagu <body bgcolor = "Aqua"> atribut *bgcolor* definiše boju pozadine stranice. Naslovi se pišu omeđeni tagovima <hn> i </hn>, gde n može imati vrednosti 1, 2, .., 6. Veće n znači manju veličinu fonta. Atribut *align* u tagu <h2> definiše poravnanje naslova. Vrednost *center* znači da će ovaj naslov biti ispisan po sredini stranice. Tag <font> definiše osobine fonta koji se koristi za ispisivanje teksta. Atribut *color* u okviru taga <font> definiše boju fonta, dok atribut *size* definiše veličinu fonta. Paragraf teksta je omeđen tagovima i . Za nastavak ispisivanja teksta u novom redu koristi se tag <br>.



Sl. 6 Primena jednostavne veb stranice

```
<html>
  <head>
    <title>ESP32-S3 WiFi Web Server</title>
    <link rel="icon" href="data:,">
  </head>
  <body bgcolor = "Aqua">
    <h2 align=center><font color="red">Sistem za navodnjavanje</font></h2>
    <hr>
    <p align=center>
    <font size=4>Nivo vode u bunaru = <b>5.25 m</b></font>
    </p><hr>
  </body>
</html>
```

Sl. 7 Jednostavna veb stranica i njen HTML kôd

U L. 2 je dat program jednostavnog veb servera. Svrha ovog programa je da na zahtev veb klijenta šalje veb stranicu sa informacijom o nivou vode u bunaru. U deklarativnom delu programa treba uočiti sledeće: Dobro poznati port protokola HTTP je 80 (linija 8). Za TCP komunikaciju koriste se klase *WiFiServer* i *WiFiClient* (linije 13 i 14). U liniji 16 deklarisan je niz *buffer* dužine 256 karaktera koji će služiti kao bafer za smeštanje karaktera primljenih preko TCP konekcije. Promenljiva k je indeks u ovom baferu, odnosno pozicija u baferu u koju će biti upisan sledeći karatker. Efekat postavljanja $k = 0$ je isto što i pražnjenje bafera.

TCP server se startuje u funkciji *setup*(), linija 42, nakon uspostavljene WiFi konekcije. Posle ove naredbe, možemo smatrati da je veb server pokrenut, odnosno da očekuje konektovanje. Program u funkciji *loop*() je organizovan u vidu konačnog automata. U stanju DISCONNECTED, ne postoji TCP konekcija. Naredba *sever.available*() služi za proveru uspostavljene konekcije. Kada se detektuje uspostavljena konekcija (*client != NULL*), prijemni bafer se prazni i prelazi se u stanje CONNECTED. U

stanju CONNECTED se čitaju primljeni karakteri HTTP zahteva i smeštaju u bafer *buffer*. Iako HTTP zahtev može imati više linija, od interesa je samo prva koja sadrži metod HTTP zahteva. Naš veb server odgovara samo na metod GET. Ako postoje primljeni, a još uvek nepreuzeti karakteri (*client.available*() > 0), čita se sledeći primljeni karakter (linija 61). Za detektovanje kraja linije, 0x0D 0x0A, potrebno je ispitati vrednost upravo pročitanog i prethodno pročitanog karaktera (linija 62). Ako nije detektovan kraj linije, pročitani karakter se upisuje u bafer i ujedno pamti u promenljivoj *lastRxChar* (linije 65 i 66), kako bi se koristio za proveru kraja linije pri čitanju sledećeg karaktera. Nakon detektovanja kraja linije, primljena linija se završava znakom za kraj stringa (koji zapravo prepisuje znak 0x0D) – linija 69. Funkcija *strstr*() u liniji 73, proverava da li string sadržan u prijemnom baferu sadrži reč GET. Ako to jeste slučaj, pozivom funkcije *sendResponse*() klijentu se šalje HTTP odziv, a onda se prelazi u stanje DISCONNECTING radi zatvaranje konekcije. Ukoliko primljena linija ne sadrži reč GET, slanje odziva izostaje i odmah se ide na zatvaranje konekcije. Ukoliko se konekcija raskine tokom prijema prve linije HTTP zahteva (proverava se funkcijom *client.connected*() u liniji 59), dalji prijem se prekida i prelazi se u stanje DISCONNECTING (linija 85). Ovakav prekid konekcije može biti samo rezultat neke abnormalne situacije. U stanju DISCONNECTING, poništavaju se svi do tog momenta primljeni, a nepročitani podaci od klijenta (linija 88) i poziva se naredba *client.stop*(), koja raskida konekciju sa klijentom. Kada se funkcijom *connected*() ustanovi da konekcija više ne postoji, FSM se vraća u stanje DISCONNECTED.

Funkcija *sendResponse*(), linije 97 – 117, šalje HTTP odziv, koji sadrži HTML veb stranice. Odziv je u najvećem delu fiksan, a jedino se menja podatak o nivou vode (linija 115).

L. 2. Program *WebServer*

```
1    #include <WiFi.h>
2    //Ultrasound:
3    const int TRIG_PIN = 39;  //triger pin ultrazvucnog modula
4    const int ECHO_PIN = 38;  //echo pin ultrazvucnog modula
5    //WiFi:
6    char ssid[] = "xxxx";     //ime WiFi mreze
7    char password[] = "xxxx"; //WiFi lozinka
8    const unsigned int HTTP_PORT = 80;  //port veb servera
9    //FSM:
10   enum FsmStateType {DISCONNECTED, CONNECTED, DISCONNECTING};
11   FsmStateType fsmState = DISCONNECTED;
12   //TCP server i klijent:
13   WiFiServer server(HTTP_PORT);
14   WiFiClient client;
15   //Prijemni bafer i indeks u prijemnom baferu
16   char buffer[256];
17   int k = 0;
18   //dva poslenje primljena karaktera:
19   char rxChar;
20   char lastRxChar;
21
22   //merenje udaljenosti: vraca izmereno rastojanje u cm
23   long range(){
24     digitalWrite(TRIG_PIN, LOW);
25     delayMicroseconds(2);
26     digitalWrite(TRIG_PIN, HIGH);
27     delayMicroseconds(10);
28     digitalWrite(TRIG_PIN,LOW);
29     long duration = pulseIn(ECHO_PIN,HIGH);
30     return duration/58;
31   }
32
```

```
33  void setup(){
34    Serial.begin(115200);
35    pinMode(TRIG_PIN, OUTPUT); //trig pin ultrazvucnog modula je izlaz
36    pinMode(ECHO_PIN, INPUT);  //echo pin ultrazvucnog modula je ulaz
37    WiFi.begin(ssid, password);
38    while (WiFi.status() != WL_CONNECTED) delay(300);
39    Serial.println("WiFi connected");
40    while (WiFi.localIP() == INADDR_NONE) delay(300);
41    Serial.print("IP address : "); Serial.println(WiFi.localIP());
42    server.begin();  //startovanje veb servera
43    Serial.println("Web server started!");
44  }
45
46  void loop(){
47    switch(fsmState){
48      case DISCONNECTED:
49        //cekanje na uspostavljanje konekcije
50        client = server.available();
51        if(client){
52          k = 0;  //praznjenje prijemnog bafera
53          lastRxChar = 0;
54          fsmState = CONNECTED;
55        }
56        break;
57
58      case CONNECTED:
59        if(client.connected()){
60          if(client.available() > 0){
61            rxChar = client.read();
62            if(rxChar != '\n' || lastRxChar != '\r'){
63              //primljeni karakteri se gomilaju u baferu sve dok se
64              //ne primi oznaka za novi red CR LF:
65              buffer[k++] = rxChar;
66              lastRxChar = rxChar;
67            } else{
68              //primljena prva linija HTTP zahteva
69              buffer[k-1] = '\0';
70              Serial.println(buffer);
71              //ako linija pocinje sa GET, saljemo HTTP odgovor
72              //i idemo na raskidanje veze:
73              if(strstr(buffer,"GET")){
74                sendResponse();
75              }
76              fsmState = DISCONNECTING;
77            }
78          }
79        }
80        else
81          //konekcija je neplanirano prekinuta:
82          fsmState = DISCONNECTING;
83        break;
84
85      case DISCONNECTING:
86        //raskidanje konekcije
87        if(client.connected()){
88          client.flush();
89          client.stop();
90        } else
91          fsmState = DISCONNECTED;
92      break;
93    }
```

```
94   }
95
96   //slanje HTTP odgovora:
97   void sendResponse(){
98     float level = ((float)range()) / 100;
99
100    //HTTP metod i zaglavlja:
101    client.println("HTTP/1.1 200 OK"); //statusna linija
102    client.println("Content-type:text/html");
103    client.println("Connection:close");
104    client.println("Refresh: 5");
105    client.println(); //prazna linija
106    //HTML:
107    client.print("<html><head>");
108    client.print("<link rel=\"icon\" href=\"data:,\">");
109    client.print("<title>ESP32-S3 WiFi Web Server</title></head>");
110    client.print("<body bgcolor=\"Aqua\">");
111    client.print("<h2 align=center><font color=\"red\">");
112    client.print("Sistem za navodnjavanje</font></h2><hr>");
113    client.print("<p align=center><font size=4>");
114    client.print("Nivo vode u bunaru = <b>");
115    client.print(level);
116    client.print("m</b></font></p><hr></body></html>");
117  }
```

Glavna prednost protokola HTTP u odnosu na komunikaciju neposredno preko TCP-a, kao u primeru iz listinga L. 1, sastoji se u tome što nije potrebno razvijati posebnu aplikaciju za prikaz rezultata na strani klijenta, jer se za prikaz veb stranice može koristiti standardni veb pregledač.

Međutim, HTTP, u obliku kako je iskorišćen u prethodnom primeru, omogućava samo jednosmernu komunikaciju, tj. prenos podataka od embeded uređaja ka klijentu. Postavlja se pitanje da li je moguće HTTP iskoristiti za prenos komandi od klijenta (pregledača) do embeded uređaja. Jedan način kako se ovo može postići je da se komande zadaju u vidu hiperveza koje umesto imena fajlova sadrže imena komandi. Nakon prijema HTTP zahteva, veb server izdvaja iz prve linije ime fajla, tj. ime komande, i izvršava je. Za realizaciju ovog koncepta potrebne su dve modifikacije prethodnog programa. Prvo, veb stranicu je potrebno proširiti komandama. Drugo, deo programa koji se bavi interpretiranjem HTTP zahteva treba proširiti kôdom za izdvajanje komandi iz metoda GET i njihovo izvršenje.

Na Sl. 8 je prikazana veb stranica sa dodatim dugmadima, PUMPAJ i STOP, pomoću kojih je moguće uključiti, odnosno isključiti pumpu. Na istoj slici vidimo HTML ove stranice. Dugme se prikazuje tagom:

```
<button> natpis </button>
```

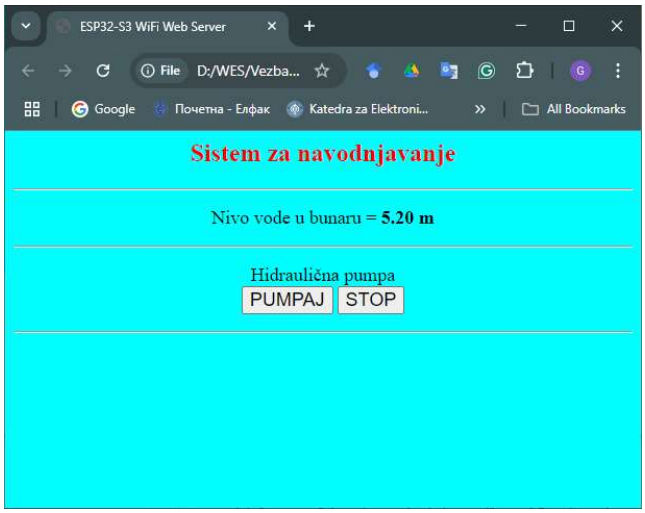
Ako želimo da klik na dugme izazove neku akciju, potrebno je tag *button* proširiti atribudom *onclick*. Vrednost ovog atributa definiše URL stranice koja će biti učitana u pregledač nakon klika na dugme. Na primer, u HTML segmentu:

```
<button onclick="location.href='/PUMPAJ'"><font size=4>PUMPAJ</font></button>
```

onclick="location.href='/HIGH'" znači da će klikom na dugme, sa natpisom HIGH, na adresu veb servera biti poslat zahtev za stranicom imena /HIGH. Slično,

```
<button onclick="location.href='/STOP"><font size=4>STOP</font></button>
```

kreira dugme sa natpisom STOP. Nakon klika na ovo dugme, pregledač će od veb servera zatražiti stranicu imena /STOP.




```
<html>
 <head>
  <title>ESP32-S3 WiFi Web Server</title>
 </head>
 <body bgcolor = "Aqua">
  <h2 align=center><font color="red">Sistem za navodnjavanje</font></h5>
  <hr>
  <p align=center>
   <font size=4>Nivo vode u bunaru = <b>5.20 m</b></font>
  </p><hr>
  <p align=center>
   <font size=4>Hidraulična pumpa</font><br>
   <button onclick="location.href='/PUMPAJ'">
   <font size=4>PUMPAJ</font></button>
   <button onclick="location.href='/STOP'">
   <font size=4>STOP</font></button>
  </p><hr>
 </body>
</html>
```

Sl. 8 Veb stranica sistema za navodnjavanje proširena dugmadima za upravljanje pumpom

Reakcija veb servera na HTTP zahteva koji nose /PUMPAJ ili /STOP je uključenje, odnosno isključenje pumpe, dok će vraćena stranica u svakom slučaju biti ista. U listingu L. 3 vidimo modifikaciju programa veb servera. U odnosu na program iz listinga L. 2, dodatni kôd postoji u stanju CONNECTED i u funkciji *sendResponse*(). Nakon slanja veb stranice, dodatno se proverava da li GET linija sadrži ime jedne od dve podržane komande. Ako u liniji postoji komanda, ona se odmah izvršava.

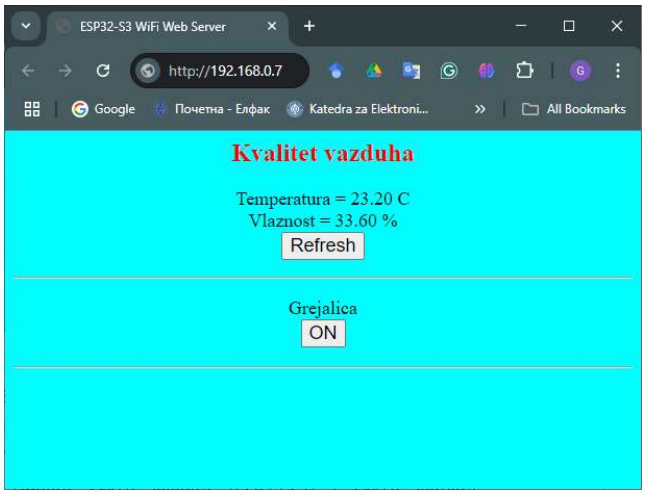
L. 3. Program *WebServer* (interpretacija komandi)

```
        case CONNECTED:
        ...
          if(strstr(buffer,"GET")){
            sendResponse();
            if(strstr(buffer,"/PUMPAJ ")){
              digitalWrite(PUMP_PIN,HIGH);
            }else if(strstr(buffer,"/STOP ")){
              digitalWrite(PUMP_PIN,LOW);
            }
          }
        ...
    void sendResponse(){
```

```
    float level = ((float)range()) / 100;

    //HTTP metod i zaglavlja:
    client.println("HTTP/1.1 200 OK"); //statusna linija
    client.println("Content-type:text/html");
    client.println("Connection:close");
    client.println("Refresh: 5");
    client.println(); //prazna linija
    //HTML:
    client.print("<html><head>");
    client.print("<link rel=\"icon\" href=\"data:,\">");
    client.print("<title>ESP32-S3 WiFi Web Server</title></head>");
    client.print("<body bgcolor=\"Aqua\">");
    client.print("<h2 align=center><font color=\"red\">");
    client.print("Sistem za navodnjavanje</font></h2><hr>");
    client.print("<p align=center><font size=4>");
    client.print("Nivo vode u bunaru = <b>");
    client.print(level);
    client.print(" m</b></font></p><hr>");
    client.print("<p align=center><font size=4>");
    client.print("Hidraulicna pumpa</font><br>");
    client.print("<button onclick=\"location.href='/PUMPAJ'\">");
    client.print("<font size=4>PUMPAJ</font></button>");
    client.print("<button onclick=\"location.href='/STOP'\">");
    client.print("<font size=4>STOP</font></button></p><hr>");
    client.print("</body></html>");
}
```

2.3 Zadatak

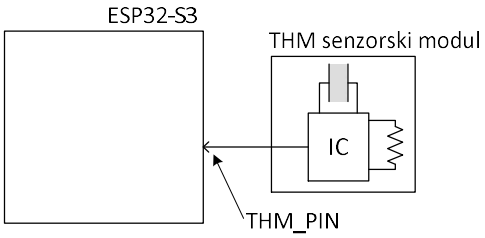
Zadatak 2.1 (20 poena): Vaš zadatak u ovoj vežbi je da realizujete embeded sistem sa veb interfejsom na bazi ESP32-S3 za monitoring i upravljanje sistemom grejanja. Sistem treba da omogući očitavanje trenutnih vrednosti temperature i vlažnosti, kao i ON/OFF upravljanje grejačem (emulirati pomoću LED). Izgled veb stranica ovog sistema je prikazan na Sl. 9. Klikom na dugme „Refresh“, stranica se osvežava (tj. ponovo učitava). Za upravljanje grejačem postoji samo jedno dugme. Klikom na ovo dugme, grejač se uključuje a natpis na dugmetu se menja na „OFF“. Ponovi klik na isto dugme, isključuje grejač, a natpis se vraća na „ON“.



Sl. 9 Veb stranica sistema grejanja

Napomena 1: Na Sl. 10(a) je prikazana fotografija senzora DTH11. Priključci „+“ i „-“ se koriste za napajanje. „+“ povezati na 3V3, a „-“ na GND. Pogrešno povezivanja napajanja može trajno da ošteti senzor! Središnji priključak „out“ se koristi za komunikaciju sa EPS32-S3. U listingu L.4 je dat program za periodično očitavanje i štampanje vlažnosti i temperature. Za rad sa senzorom DHT11 koristi se klasa DHT11.

(a)



(b)

Sl. 10 Senzor temperature i vlažnosti DTH11: (a) fotografija senzorskog modula; (b) sprega sa mikrokontrolerom

L. 4. Program *DHT11*

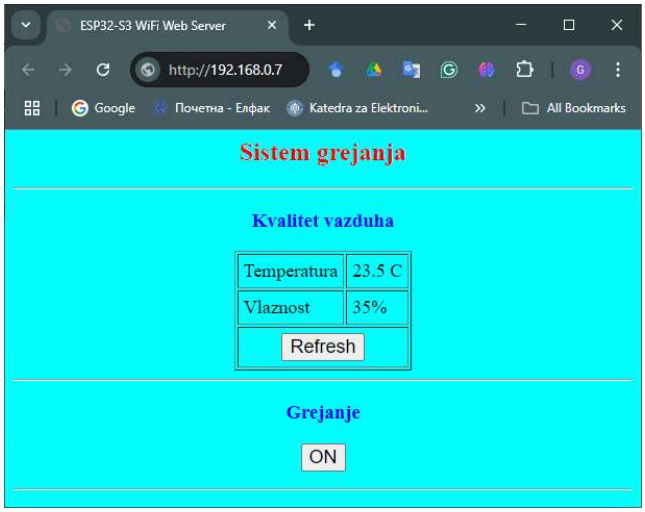
```
#include "DHT11.h"
const int THM_PIN = 2;  //pin TH modula
//objekat klase THM i poslednje izmerena temperatura i vlaznost
DHT11 dht11(THM_PIN);
float temperature;
float humidity;

void setup() {
  Serial.begin(115200);
}

void loop() {
  temperature = dht11.readTemperature();
  humidity = dht11.readHumidity();

  Serial.print("T = ");Serial.print(temperature);Serial.print(" C");
  Serial.print("  H = ");Serial.print(humidity);Serial.println(" %");

  delay(1000);
}
```

Napomena 2: Umesto da HTML stranicu pišete direktno u programu pomoću naredbi *client.print*(), preporučuje se da najpre kreirate stranicu u običnom tekstualnom editoru (npr. Notepad) i sačuvate je sa ekstenzijom .htm. Tako kreirani fajl možete otvoriti duplim klikom u veb pregledaču i proveriti izgled stranice. U ovoj fazi koristite fiksne (probne) vrednosti za temperaturu i vlažnost. Kada budete zadovoljni izgledom i rasporedom elemenata, prebacite HTML kod u program i na odgovarajućim mestima zamenite fiksne vrednosti stvarnim promenljivama koje prikazuju aktuelne podatke (temperaturu i vlažnost).

Napomena 3: Za razliku od sistema za navodnjavanje, u ovom slučaju postoji samo jedno dugme za upravljanje uređajem. To znači da će se ka ESP32-S3 modulu slati samo jedna komanda, na primer, TOGGLE, koja će promeniti trenutno stanje odgovarajućeg ON/OFF signala. Takođe, prilikom generisanja HTML stranice, ESP32-S3 bi trebalo da na dugmetu prikaže odgovarajući natpis u skladu sa trenutnim stanjem: „ON“ ako je uređaj uključen, odnosno „OFF“ ako je isključen.

Napomena 4: Za razliku od sistema za navodnjavanje, gde se veb stranica automatski osvežavala svakih 5 sekundi, zahvaljuju `client.println("Refresh: 5")` u liniji 104 iz listinga L.2, veb stranica sistema grejanja se ne osvežava automatski (što znači da navedenu liniju programa treba izostaviti) već klikom na dugme Refresh.

Zadatak 2.2 (30 poena): Modifikovati veb stranicu sistema grejana prema prikazu na Sl. 11. Izmena je samo u HTML-u, u delu koji se odnosi na podatke o kvalitetu vazduha koji su sada prikazani u tabeli. Takođe, natipisi „Sistem grejanja“, „Kvalitet vazduha“ i „Grejanje“ su formatirani kao naslovi. Progamska logika treba da ostane nepromenjena.



Sl. 11 Modifikovana veb stranica sistema grejanja

Napomena 5: Informacije o tome kako se kreiraju HTML table možete videti u pratećem materijalu, ili potražiti na Internetu.

- ♦ ♦ ♦ -