Tehnička dokumentacija
za pametan parking

Mentor: Dr. Nebojša Andrijević

Asistent: Nikola Jović

Student: Nikola Pejak

Indeks: SI-301-20

Novi Sad,

Septembar, 2024.

SADRŽAJ:

1. Uvod

Ovaj dokument opisuje tehničke specifikacije i funkcionalnosti pametnog parking sistema zasnovanog na Arduino Mega mikrokontroleru, ESP8266 WiFi modulu, DFPlayer Mini MP3 modulu i DHT11 senzorima za monitoring temperature. Sistem omogućava rezervaciju parking mesta putem web stranice i automatski kontroliše pristup vozila, pištanje alarma u slučaju požara, kao i rad ventilatora za održavanje temperature

2. Komponente Sistema

- **Arduino Mega 2560**: Centralni kontroler za upravljanje celokupnim sistemom.
- **ESP8266 WiFi Modul**: Za povezivanje sa WiFi mrežom i komunikaciju sa web stranicom.
- **DFPlayer Mini MP3 Modul**: Za reprodukciju audio zapisa i upozorenja.
- **DHT11 Senzor**: Merenje temperature i vlažnosti u parking prostoru.
- **Ultrazvučni Senzori**: Za detekciju prisustva vozila na parking mestima.
- **IR Senzori**: Za detekciju ulaska i izlaska vozila sa parkinga.
- **Ventilator**: Kontrola temperature unutar parkinga.
- **LCD Displej ST7920 (128x64)**: Prikaz informacija o slobodnim/zauzetim parking mestima, temperaturi, i statusu sistema.
- **Servo Motor**: Kontroliše rampu za ulazak i izlazak vozila.
- **Releji**: Za kontrolu ventilatora i drugih uređaja.

3. Arduino Mega R3 2560

Arduino Mega 2560 R3 je mikrokontrolerska ploča zasnovana na ATmega2560 mikrokontroleru. Ova ploča je popularna zbog svoje velike količine ulaza/izlaza, memorije i brzine, što je čini odličnim izborom za složenije projekte sa više senzora, motora i komunikacionih modula.

- **Osnovne Karakteristike:**

1. **Mikrokontroler:** ATmega2560
2. **Radni napon:** 5V
3. **Ulazni napon (preko napajanja):** 7-12V
4. **Digitalni I/O pinovi: 54 (od kojih 15 može da bude PWM izlaz)**
5. **Analogni ulazi: 16**
6. **PWM kanali: 15**
7. **Serijski portovi: 4 (Serial0, Serial1, Serial2, Serial3)**
8. **Flash memorija: 256 KB (od toga 8 KB zauzima bootloader)**
9. **SRAM: 8 KB**
10. **EEPROM: 4 KB**
11. **Takt procesora: 16 MHz**
12. **USB konektor: Koristi USB konekciju za programiranje i napajanje ploče.**

- **Prednosti Arduino Mega 2560 R3:**

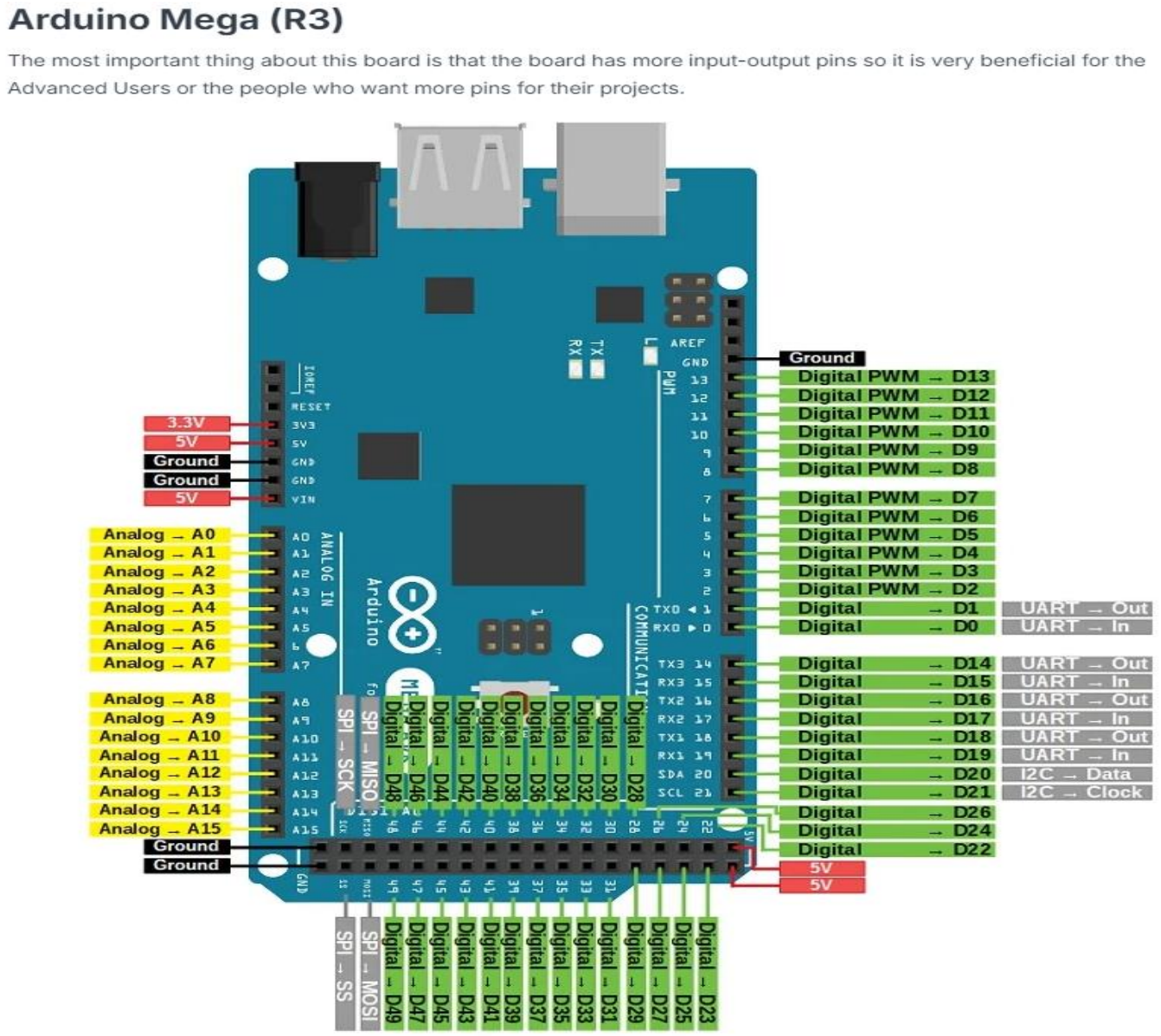
1. **Veliki broj pinova:** Sa 54 digitalna I/O pina i 16 analognih ulaza, Arduino Mega omogućava povezivanje sa više senzora i aktuatora istovremeno.
2. **Više serijskih portova:** Ploča poseduje 4 serijska porta (TX/RX parova) što omogućava komunikaciju sa više serijskih uređaja (npr. GPS, Wi-Fi moduli, MP3 plejeri) bez potrebe za softverskim serijskim portovima.
3. **Veća memorija:** Arduino Mega ima više memorije u odnosu na druge Arduino ploče kao što su Arduino Uno ili Nano, što omogućava kompleksnije programe i više prostora za promene i dodavanje funkcionalnosti.
4. **Kompatibilnost:** Arduino Mega 2560 R3 je kompatibilan sa većinom Arduino dodataka (shieldova) i softverskih biblioteka, što ga čini univerzalnim za različite projekte.
5. **Složeniji projekti:** Ploča je pogodna za projekte koji zahtevaju više simultanih funkcionalnosti, kao što su roboti, pametni sistemi (npr. pametni parking) i komunikacioni projekti.

➢ **Upotreba u Projektima:**

Zbog velikog broja ulaznih i izlaznih pinova, Arduino Mega se često koristi u složenim projektima kao što su:

- **Robotika:** Upravljanje više motora i senzora.
- **Pametni sistemi:** Kao što je tvoj pametni parking sistem, gde je potrebno upravljanje raznim senzorima, ekranima, relejima i komunikacionim modulima.
- **Sistemi za prikupljanje podataka:** Ploča može upravljati sa više analognih i digitalnih senzora za prikupljanje podataka.
- **Upravljanje displejima:** Kao kod korišćenja sa LCD i OLED ekranima za interfejs sistema.

Arduino Mega je jedan od najmoćnijih članova Arduino porodice i koristi se tamo gde osnovne ploče, poput Uno, nemaju dovoljno resursa za složenije aplikacije.



4. Funkcionalnosti Sistema

- **Rezervacija Parking Mesta**: Kroz web interfejs korisnik može rezervisati parking mesto.
- **Detekcija Prisustva Vozila**: Korišćenje ultrazvučnih senzora za monitoring zauzetih i slobodnih parking mesta.
- **Reprodukcija Upozorenja**: Reprodukcija audio zapisa putem DFPlayer Mini
- **Monitoring Temperature**: Praćenje temperature i automatska kontrola ventilatora kada temperatura pređe zadatu vrednost.
- **Automatska Kontrola Rampe**: Otvaranje i zatvaranje rampe pomoću servo motora na osnovu detekcije vozila i tastera za ulaz.
- **Piezo upozorenje**: za upozorenje vozača ili u slučaju detekcije požara
- **Senzor vode sa bunaro**: služi kao rezervoar za vodenu pumpu
- **Vakum puma**: za gašenje požara
- **Ventilacija na ključ prekidač**: Za provetravanje parkinga
- **Otvaranje rampe putem sajta**
- **Zatvaranje rampe putem sajta**
- **Puštanje muzike putem sajta**

4. Softverska Implementacija

4.1. Arduino Program

➢ Biblioteke:

- DFRobotDFPlayerMini.h za MP3 reprodukciju
- DHT.h za rad sa DHT11 senzorom
- U8g2lib.h za LCD ST7920 displej
- Servo.h za kontrolu servo motora
- SoftwareSerial.h za komunikaciju sa ESP8266 (ako je potrebno

```
1. #include <DHT.h>
2. #include <DHT_U.h>
3. #include "DFRobotDFPlayerMini.h"
4.
5. #include <DHT11.h>
6.
7. #include <SoftwareSerial.h> // SoftwareSerial library
8.
9. #include <LiquidCrystal_I2C.h>
10.
11.#define servoPin 44
12.
13.#include <U8g2lib.h>
14.#include <Servo.h>
15.

16.#define trigPin_3 43
17.#define echoPin_3 42
18.









19.Servo ServoRampa;
20.
21.DFRobotDFPlayerMini myDFPlayer;
22.

23.#define DHTPIN 42     // Pin na koji je povezan DHT senzor
24.#define DHTTYPE DHT11   // Tip senzora (DHT11 ili DHT22)
25.
26.DHT dht(DHTPIN, DHTTYPE);  // Kreiramo instancu DHT senzora
```

```
27.
28.int rxPin = 17; // RX2 pin na Arduino Mega
29.int txPin = 16; // TX2 pin na Arduino Mega
30.
31.bool alarmAktiviran = false;
32.

33.int IR_Ulaz = 6;
34.int IR_Izlaz = 7;
35.
36.int ParkingMesta = 3;
37.int ZauzetaMesta = 0;
38.
39.int Prvo_Stanje = 0;
40.int Drugo_Stanje = 0;
41.
42.int piezoPin = 26;
43.
44.long distance;
45.



46.long distance_2;
47.
48.long distance_3;
49.
50.int buttonUlaz = 4;
51.
52.int buttonState = 0;  // Trenutno stanje tastera
53.
54.int lastButtonState = 0;  // Prethodno stanje tastera
55.
56.bool isOpen = false;
57.
58.bool carDetected = false;
59.bool carDetected2 = false;
60.
61.const int AnalogWaterPin = A5;
62.
63.const int SemaforCrvno_1 = 22;
64.const int SemaforZuto_1 = 23;
65.const int SemaforZeleno_1 = 24;
66.
67.const int SemaforCrvno_2 = 38;
68.const int SemaforZuto_2 = 39;
69.const int SemaforZeleno_2 = 40;
70.
71.const int SenzorPlamena = A0;
72.const int pragAktivacije = 60; // Prag za aktivaciju alarma
```

```
73.
74. const int PumpPin = A2; // Pin na koji je povezan prekidač
75. const int relayPin = A4;
76.
77. const int VentilatorPin = A8;
78. const int ReleyPinFan = A9;
79.
80. bool espConnected = false;  // Promenljiva koja prati status konekcije ESP8266
81.

82. enum StanjeSemafora {
83.   CRVENO,
84.   ZELENO,
85.   BLINK_ZELENO,
86.   ZUTO_1,
87.   ZUTO_2
88. };
89.
90. StanjeSemafora trenutnoStanje = CRVENO;
91.
92. StanjeSemafora trenutnoStanje2 = ZELENO;
93.
94. unsigned long prethodniMillis = 0;
95.
96. unsigned long prethodniMillis2 = 0;
97.
98. const long trajanjeCrvenog = 7000;  // Trajanje crvenog svetla u milisekundama
99. const long trajanjeZelenog = 7000;  // Trajanje zelenog svetla u milisekundama
100.      const long trajanjeBlinkanja = 4000; // Trajanje blinkanja zelenog svetla u
    milisekundama
101.      const long trajanjeZutog = 2000;    // Trajanje žutog svetla u milisekundama
102.
103.      unsigned long blinkMillis = 0;  // Vreme za blinkanje zelenog svetla
104.      bool zelenoBlinkStanje = false; // Trenutno stanje blinkanja zelenog svetla
105.      //------------------------------------------------------------------------
    --------------------------
106.      bool zelenoBlinkStanje2 = false; // Trenutno stanje blinkanja zelenog svetla
    drugog semafora
107.
108.      unsigned long blinkMillis2 = 0;  // Vreme za blinkanje zelenog svetla drugog
    semafora
109.
110.      // Definišemo minimalnu i maksimalnu udaljenost (u centimetrima)
111.      const int minDistance = 5;    // Minimalna udaljenost (najbliže senzoru)
112.      const int maxDistance = 10;  // Maksimalna udaljenost (najdalje od senzora)
113.
114.      // Ako koristite ST7920 sa SPI interfejsom
115.      U8G2_ST7920_128X64_F_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /*
    CS=*/ 10, /* reset=*/ 8);
116.
```

```
117.    void setup()
118.     {
119.
120.      dht.begin();  // Inicijalizacija DHT senzora
121.      Serial.begin(9600);    // Serijska komunikacija sa računarom
122.      Serial2.begin(115200); // Serijska komunikacija sa ESP8266 (RX2/TX2)
123.      Serial.println("Arduino Mega ready");
124.
125.      Serial2.println("Are you there, ESP8266?"); // Šalji proveru na ESP8266
126.
127.      pinMode(PumpPin, INPUT_PULLUP);
128.      pinMode(relayPin, OUTPUT);
129.      digitalWrite(relayPin, LOW);       // Početno stanje releja je isključeno
130.
131.
132.
133.      pinMode(VentilatorPin,INPUT_PULLUP);
134.
135.      pinMode(ReleyPinFan, OUTPUT);
136.      digitalWrite(ReleyPinFan, LOW);
137.
138.     pinMode(buttonUlaz, INPUT_PULLUP);
139.
140.      pinMode(31, OUTPUT);  // Trig pin kao izlaz
141.      pinMode(30, INPUT);   // Echo pin kao ulaz
142.
143.      pinMode(50, OUTPUT);  // Trig pin kao izlaz
144.      pinMode(51, INPUT);   // Echo pin kao ulaz
145.
146.      pinMode(45, OUTPUT);  // Trig pin kao izlaz
147.      pinMode(44, INPUT);   // Echo pin kao ulaz
148.      u8g2.begin(); // Inicijalizacija displeja
149.      u8g2.setContrast(255); // Postavljanje maksimalnog kontrasta
150.      pinMode(IR_Ulaz, INPUT);
151.      pinMode(IR_Izlaz, INPUT);
152.
153.      pinMode(SemaforCrvno_1,OUTPUT);
154.      pinMode(SemaforZuto_1,OUTPUT);
155.      pinMode(SemaforZeleno_1,OUTPUT);
156.
157.      digitalWrite(SemaforCrvno_1, LOW);
158.      digitalWrite(SemaforZuto_1, LOW);
159.      digitalWrite(SemaforZeleno_1, LOW);
160.
161.      pinMode(SemaforCrvno_2, OUTPUT);
162.      pinMode(SemaforZuto_2, OUTPUT);
163.      pinMode(SemaforZeleno_2, OUTPUT);
```

```
164.
165.        digitalWrite(SemaforCrvno_2, LOW);
166.        digitalWrite(SemaforZuto_2, LOW);
167.        digitalWrite(SemaforZeleno_2, LOW);
168.
169.        pinMode(SenzorPlamena,INPUT);
170.
171.
172.
173.        trenutnoStanje = CRVENO;
174.        prethodniMillis = millis();
175.
176.        trenutnoStanje2 = ZELENO;
177.        prethodniMillis2 = millis();
178.
179.        ServoRampa.attach(servoPin);
180.        ServoRampa.write(0);
181.
182.      }
183.
184.      void loop()
185.      {
186.        //------------------------------------------------------------------------
187.        Semafori();
188.        ProveraKonekcije();
189.        KontrolaVodenePumpe();
190.        KontrolaVentilacije();
191.        Vodostaj();
192.        kontrolaUlazaIzlaza();
193.        DHTSenzor();
194.        proveriSenzorPlamena();
195.
196.      //-----------------------------------NIVO VODE---------------------------------
    ---------------------------------
197.
198.      //-----------------------------------------PARKING SENZOR 1-----------------
    ----------------------------------------
199.      long distance = readUltrasonic(31, 30);
200.      long distance_2 = readUltrasonic(50, 51);
201.        controlBuzzer(distance);
202.        delay(100); // Mala pauza između merenja
203.
204.        controlBuzzer(distance_2);  // Kontroliše pištanje za drugi senzor
205.          delay(100); // Mala pauza između merenja
206.
207.        delay(100); // Kratka pauza pre sledećeg čitanja
208.
```

```
209.        if (distance < 5)
210.        {
211.          if (!carDetected)
212.          {
213.            carDetected = true;  // Označi da je auto detektovan
214.            if (ParkingMesta > 0)
215.            {
216.              ParkingMesta--;  // Smanji broj slobodnih parking mesta
217.              ZauzetaMesta++;
218.            }
219.          }
220.        }
221.        else
222.        {
223.          if (carDetected)
224.          {
225.            carDetected = false;  // Resetuj detekciju kada auto ode
226.            ParkingMesta++;
227.            ZauzetaMesta--;
228.          }
229.        }
230.
231.
232.        if (distance_2 < 5)
233.        {
234.          if (!carDetected2)
235.          {
236.            carDetected2 = true;  // Označi da je auto detektovan
237.            if (ParkingMesta > 0)
238.            {
239.              ParkingMesta--;  // Smanji broj slobodnih parking mesta
240.              ZauzetaMesta++;
241.            }
242.          }
243.        }
244.        else
245.        {
246.          if (carDetected2)
247.          {
248.            carDetected2 = false;  // Resetuj detekciju kada auto ode
249.            ParkingMesta++;
250.            ZauzetaMesta--;
251.          }
252.        }
253.      //-----------------------------------------------
254.      //---------------SPUŠTANJE RAMPE-----------------------------
255.      //-----------------------------------------------
256.      if(Prvo_Stanje==1 && Drugo_Stanje==1)
257.      {
258.          ZatvoriRampu();
```

```
259.          Prvo_Stanje=0, Drugo_Stanje=0;
260.      }
261.      //---------------------------------------------
262.        u8g2.clearBuffer();          // Brisanje bafera
263.        u8g2.setFont(u8g2_font_ncenB08_tr); // Postavljanje fonta
264.        u8g2.setCursor(0, 10);
265.        u8g2.print("Slob. mesta: ");
266.        u8g2.print(ParkingMesta);
267.        u8g2.setCursor(0, 24);
268.        u8g2.print("Zauz. mesta: ");
269.        u8g2.print(ZauzetaMesta);
270.      //-----------------------PARKINZI NA DISPLEJU
271.        u8g2.setCursor(0, 35);
272.        u8g2.print("P1: ");
273.        u8g2.print(distance);
274.
275.        u8g2.setCursor(55, 35);
276.        u8g2.print("P2: ");
277.        u8g2.print(distance_2);
278.
279.        u8g2.setCursor(55, 45);
280.        u8g2.print("P3: ");
281.        u8g2.print(distance_3);
282.      //---------------------------------------------------------------------------
    ---------------------
283.
284.      //---------------------------------------------------------------------------
    --------------------
285.      }
286.
287.      //-------------------------------------------------------------------VOIDI
288.      void proveriSenzorPlamena() {
289.        int stanjeVatre = analogRead(SenzorPlamena); // Čitanje vrednosti sa
    senzora plamena
290.        Serial.println(stanjeVatre); // Ispisuje vrednost senzora na serijski
    monitor za debagovanje
291.
292.        if (stanjeVatre < pragAktivacije) { // Ako je vrednost ispod praga (požar
    detektovan)
293.          tone(piezoPin, 1500);  // Aktivira pištanje piezo zvučnika
294.        } else {
295.          noTone(piezoPin);  // Isključuje pištanje piezo zvučnika
296.        }
297.      }
298.      void DHTSenzor()
299.      {
300.       // Očitavanje temperature i vlažnosti
301.        float vlaznost = dht.readHumidity();
302.        float temperatura = dht.readTemperature();
303.
```

```
304.        // Provera da li su očitavanja uspešna
305.        if (isnan(vlaznost) || isnan(temperatura)) {
306.          Serial.println("Greska pri ocitavanju DHT senzora!");
307.          return;
308.        }
309.
310.        // Ispisivanje očitanih vrednosti na serijski monitor
311.        Serial.print("Vlaznost: ");
312.        Serial.print(vlaznost);
313.        Serial.print(" %\t");
314.        Serial.print("Temperatura: ");
315.        Serial.print(temperatura);
316.        Serial.println(" *C");
317.
318.        delay(500);  // Pauza između očitavanja
319.      }
320.      void prikaziTekst(int tekstIndex)
321.      {
322.        u8g2.clearBuffer();
323.
324.        switch (tekstIndex) {
325.          case 0:
326.            u8g2.drawStr(0, 24, "Dobrodosli!"); // Prikazuje prvi tekst
327.            break;
328.        }
329.
330.        u8g2.sendBuffer(); // Slanje bafera na displej
331.      }
332.      unsigned long prethodnoVremeUlaz = 0;
333.      const unsigned long vremeCekanja = 100; // Vreme čekanja od 100 milisekundi
    za debounce
334.      void kontrolaUlazaIzlaza()
335.      {
336.        unsigned long trenutnoVreme = millis();
337.        // Provera stanja tastera za otvaranje rampe
338.        buttonState = digitalRead(buttonUlaz); // Čitanje trenutnog stanja tastera
339.        if (buttonState != lastButtonState && (trenutnoVreme - prethodnoVremeUlaz
    >= vremeCekanja)) {
340.          if (buttonState == LOW) {
341.            OtvoriRampu();
342.            isOpen = true;
343.          }
344.          isOpen = !isOpen; // Promena stanja rampe
345.          prethodnoVremeUlaz = trenutnoVreme; // Ažuriraj vreme ulaza
346.        }
347.
348.        // Provera ulaza na parking
349.        if (digitalRead(IR_Ulaz) == LOW && Prvo_Stanje == 0)
350.        {
351.          Prvo_Stanje = 1;
```

```
352.       }
353.
354.       // Provera izlaza sa parkinga
355.       if (digitalRead(IR_Izlaz) == LOW && Drugo_Stanje == 0 && (trenutnoVreme -
   prethodnoVremeUlaz >= vremeCekanja))
356.       {
357.         Drugo_Stanje = 1;
358.         if (Prvo_Stanje == 0)
359.         {
360.           OtvoriRampu();
361.         }
362.         prethodnoVremeUlaz = trenutnoVreme; // Ažuriraj vreme izlaza
363.       }
364.

365.       lastButtonState = buttonState; // Skladištenje trenutnog stanja tastera za
   sledeću iteraciju
366.     }
367.     void Vodostaj() {
368.       int NivoVode = analogRead(AnalogWaterPin); // Čitanje nivoa vode sa
   analognog pina
369.
370.       u8g2.setCursor(0, 55); // Podešavanje kursora za prikaz na ekranu
371.       u8g2.print("Voda:  ");
372.       u8g2.print(NivoVode); // Prikazivanje nivoa vode
373.       u8g2.sendBuffer();  // Slanje bafera na displej
374.
375.       delay(100); // Mala pauza pre sledećeg osvežavanja
376.     }
377.     void KontrolaVentilacije()
378.     {
379.       if(digitalRead(VentilatorPin) == HIGH)
380.       {
381.           digitalWrite(ReleyPinFan,HIGH);
382.       }
383.       else
384.       {
385.          digitalWrite(ReleyPinFan,LOW);
386.       }
387.     }
388.       bool pumpState = false;  // Početno stanje pumpe - isključena
389.     void KontrolaVodenePumpe() {
390.
391.       int StanjePumpe = digitalRead(PumpPin);
392.       digitalWrite(relayPin, (StanjePumpe == LOW) ? HIGH : LOW);
393.     }
394.     void ProveraKonekcije()
395.     {
396.       float temperature = dht.readTemperature();
397.       if (Serial.available())
```

```
398.       {
399.         String command = Serial.readStringUntil('\n'); // Čita komandu sa
   serijskog porta
400.         command.trim(); // Uklanja eventualne razmake i prazne redove
401.         if (command == "OPEN")
402.         {
403.             OtvoriRampu();
404.         }
405.         if (command == "CLOSE")
406.         {
407.             ZatvoriRampu();
408.         }
409.
410.         if (command == "RESERVE1")
411.         {
412.           if (ParkingMesta > 0) {
413.             ParkingMesta--;  // Smanjuje broj slobodnih parking mesta
414.             ZauzetaMesta++;  // Povećava broj zauzetih parking mesta
415.             Serial.println("Parking mesto je rezervisano.");
416.           }
417.            else
418.           {
419.              tone(piezoPin, 255);
420.           }
421.         }
422.         if (command == "PLAY")
423.         {
424.             Serial.println("Pustanje prve pesme...");
425.             myDFPlayer.setTimeOut(500);  // Serijski timeout 500ms
426.             myDFPlayer.volume(25);       // Podešavanje jačine zvuka (opseg od 0
   do 30)
427.             myDFPlayer.EQ(0);            // Normalna ekvilizacija
428.             myDFPlayer.play(1);  // Pusti prvu pesmu (0001.mp3) sa SD kartice
429.         }
430.         if(command == "STOP")
431.         {
432.             myDFPlayer.stop();
433.         }
434.         if (command == "GET_STATUS") {
435.           int SlobodnaMesta = ParkingMesta - ZauzetaMesta;
436.           String statusMessage = String(SlobodnaMesta) + "," +
   String(ZauzetaMesta);
437.           Serial.println(statusMessage);  // Šaljemo ESP8266 broj slobodnih i
   zauzetih mesta
438.           Serial.println("Poslat status: " + statusMessage);  // Za debagovanje
439.         }
440.       }
441.     }
442.
443.     void Semafori()
```

```
444.     {
445.       unsigned long trenutniMillis = millis();
446.     unsigned long trenutniMillis2 = millis();
447.
448.       switch (trenutnoStanje)
449.       {
450.        case CRVENO:
451.           digitalWrite(SemaforCrvno_1, HIGH);
452.           digitalWrite(SemaforZuto_1, LOW);
453.           digitalWrite(SemaforZeleno_1, LOW);
454.           if (trenutniMillis - prethodniMillis >= trajanjeCrvenog) {
455.             trenutnoStanje = ZUTO_1;
456.             prethodniMillis = trenutniMillis;
457.           }
458.           break;
459.         case ZUTO_1:
460.           digitalWrite(SemaforCrvno_1, LOW);
461.           digitalWrite(SemaforZuto_1, HIGH);
462.           digitalWrite(SemaforZeleno_1, LOW);
463.           if (trenutniMillis - prethodniMillis >= trajanjeZutog) {
464.             trenutnoStanje = ZELENO;
465.             prethodniMillis = trenutniMillis;
466.           }
467.           break;
468.         case ZELENO:
469.           digitalWrite(SemaforCrvno_1, LOW);
470.           digitalWrite(SemaforZuto_1, LOW);
471.           digitalWrite(SemaforZeleno_1, HIGH);
472.           if (trenutniMillis - prethodniMillis >= trajanjeZelenog) {
473.             trenutnoStanje = BLINK_ZELENO;
474.             prethodniMillis = trenutniMillis;
475.             blinkMillis = trenutniMillis;
476.           }
477.           break;
478.
479.         case BLINK_ZELENO:
480.           if (trenutniMillis - blinkMillis >= 500) {
481.             blinkMillis = trenutniMillis;
482.             zelenoBlinkStanje = !zelenoBlinkStanje;
483.             digitalWrite(SemaforZeleno_1, zelenoBlinkStanje ? HIGH : LOW);
484.           }
485.           if (trenutniMillis - prethodniMillis >= trajanjeBlinkanja) {
486.             trenutnoStanje = ZUTO_2;
487.             prethodniMillis = trenutniMillis;
488.           }
489.           break;
490.
491.         case ZUTO_2:
492.           digitalWrite(SemaforCrvno_1, LOW);
493.           digitalWrite(SemaforZuto_1, HIGH);
```

```
494.            digitalWrite(SemaforZeleno_1, LOW);
495.            if (trenutniMillis - prethodniMillis >= trajanjeZutog) {
496.              trenutnoStanje = CRVENO;
497.              prethodniMillis = trenutniMillis;
498.            }
499.            break;
500.        }
501.
502.        switch (trenutnoStanje2) {
503.          case ZELENO:
504.            digitalWrite(SemaforCrvno_2, LOW);
505.            digitalWrite(SemaforZuto_2, LOW);
506.            digitalWrite(SemaforZeleno_2, HIGH);
507.            if (trenutniMillis2 - prethodniMillis2 >= trajanjeZelenog) {
508.              trenutnoStanje2 = BLINK_ZELENO;
509.              prethodniMillis2 = trenutniMillis2;
510.              blinkMillis2 = trenutniMillis2;
511.            }
512.            break;
513.
514.          case BLINK_ZELENO:
515.            if (trenutniMillis2 - blinkMillis2 >= 500) {
516.              blinkMillis2 = trenutniMillis2;
517.              zelenoBlinkStanje2 = !zelenoBlinkStanje2;
518.              digitalWrite(SemaforZeleno_2, zelenoBlinkStanje2 ? HIGH : LOW);
519.            }
520.            if (trenutniMillis2 - prethodniMillis2 >= trajanjeBlinkanja) {
521.              trenutnoStanje2 = ZUTO_1;
522.              prethodniMillis2 = trenutniMillis2;
523.            }
524.            break;
525.
526.          case ZUTO_1:
527.            digitalWrite(SemaforCrvno_2, LOW);
528.            digitalWrite(SemaforZuto_2, HIGH);
529.            digitalWrite(SemaforZeleno_2, LOW);
530.            if (trenutniMillis2 - prethodniMillis2 >= trajanjeZutog) {
531.              trenutnoStanje2 = CRVENO;
532.              prethodniMillis2 = trenutniMillis2;
533.            }
534.            break;
535.
536.          case CRVENO:
537.            digitalWrite(SemaforCrvno_2, HIGH);
538.            digitalWrite(SemaforZuto_2, LOW);
539.            digitalWrite(SemaforZeleno_2, LOW);
540.            if (trenutniMillis2 - prethodniMillis2 >= trajanjeCrvenog) {
541.              trenutnoStanje2 = ZUTO_2;
542.              prethodniMillis2 = trenutniMillis2;
543.            }
```

```
            break;
          case ZUTO_2:
            digitalWrite(SemaforCrvno_2, LOW);
            digitalWrite(SemaforZuto_2, HIGH);
            digitalWrite(SemaforZeleno_2, LOW);
            if (trenutniMillis2 - prethodniMillis2 >= trajanjeZutog) {
              trenutnoStanje2 = ZELENO;
              trenutniMillis2 = trenutniMillis2;
            }
            break;
        }
      }

      // Funkcija za spuštanje rampe
      void OtvoriRampu()
      {
        slowMove(ServoRampa, 0, 90, 10);

      }
      void ZatvoriRampu()
      {
        slowMove(ServoRampa, 90, 0, 10);
      }
      void PunParking()
      {
        u8g2.setFont(u8g2_font_ncenB08_tr); // Postavljanje fonta
        u8g2.setCursor(0, 50);
        u8g2.print("Parking je pun!");
        u8g2.sendBuffer();           // Slanje bafera na displej
        tone(piezoPin, 255);
        delay(1000);  // Pauza od 1 sekunde
        noTone(piezoPin);
      }
      //---------------------------------
      //-----------------------------------

      // Funkcija za sporo pomeranje servo motora
      void slowMove(Servo &servo, int start, int end, int delayTime)
      {
        if (start < end) {
          for (int pos = start; pos <= end; pos++) {
            servo.write(pos);  // Postavljanje pozicije servo motora
            delay(delayTime);  // Pauza za usporavanje kretanja
          }
        } else {
          for (int pos = start; pos >= end; pos--) {
            servo.write(pos);  // Postavljanje pozicije servo motora
            delay(delayTime);  // Pauza za usporavanje kretanja
          }
        }
```

```
}
long readUltrasonic(int trigPin, int echoPin)
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = (duration * 0.034) / 2;

  return distance;
}
void controlBuzzer(long distance) {
  if (distance > 0 && distance < 5) { // Podesite maksimalnu udaljenost za detekciju
    int frequency = map(distance, 0, 5, 2000, 100); // Invertovana logika: bliže objekat -> viša frekvencija
    tone(piezoPin, frequency, 100); // Pišti 100 ms
  } else {
    noTone(piezoPin); // Zaustavi pištanje ako nema detekcije ili je objekat predaleko
  }
}
//------------------------------------------------------------------------------------
//------------------------------------------------------------------------------------
void updateLCD() {
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_ncenB08_tr);
  u8g2.setCursor(0, 10);
  u8g2.print("Slob. mesta: ");
  u8g2.print(ParkingMesta);
  u8g2.setCursor(0, 24);
  u8g2.print("Zauz. mesta: ");
  u8g2.print(ZauzetaMesta);
  u8g2.sendBuffer();
}

```

5. Web Interfejs

- HTML, CSS, i JavaScript kod za interfejs koji omogućava rezervaciju parking mesta.
- ESP8266 je konfigurisan kao server i šalje podatke Arduino Mega putem serijske komunikacije.

6. Zaključak

Ova tehnička dokumentacija pruža sveobuhvatne informacije o komponentama, šemama povezivanja, funkcionalnostima i softverskoj implementaciji pametnog parking sistema. Sistem je projektovan za jednostavnu upotrebu i mogućnost proširenja za dodatne funkcionalnosti.

Topla preporuka ukoliko želite da uzmete sebi i koristite arduino megu i olakšate sebi život i krvni pritisak, kupite arduino megu 2560 sa već ugradnjeim WiFi modulom (kako ne biste imali problema i zlopatili sa dodatnim wifi modulima)