****

**Tehnička dokumentacija**

**za pametan parking**

**Mentor: Dr. Nebojša Andrijević Student: Nikola Pejak**

**Asistent:Nikola Jović Indeks: SI-301-20**

***Novi Sad,***

***Septembar, 2024.***

**SADRŽAJ:**

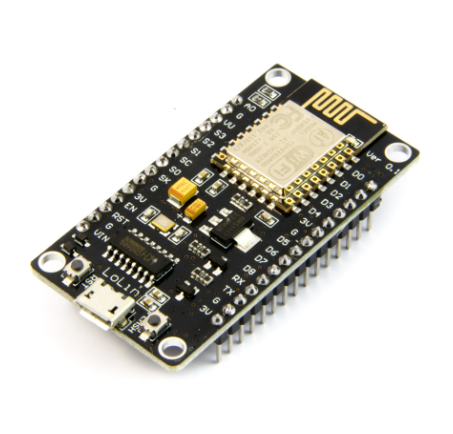
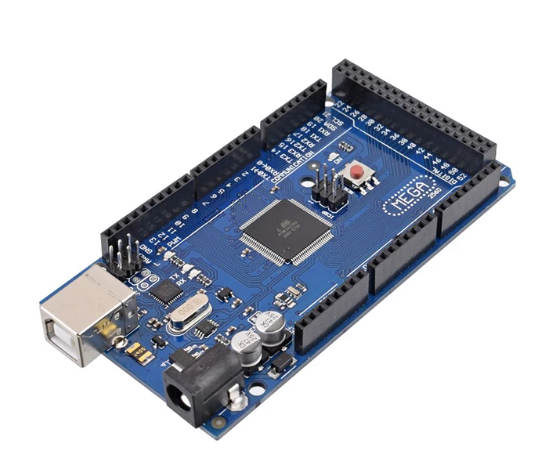
1. Uvod…………………………………………………………………………….
2. Komponente Sistema……………………………………………………………..
3. Arduino Mega R3 2560…………………………………………………………..
4. Funkcionalnosti Sistema………………………………………………………....
5. Softverska Implementacija…………………………………………………….....
6. Zaključak………………………………………………………………................

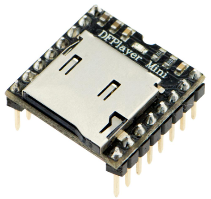
**1. Uvod**

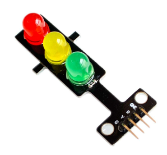
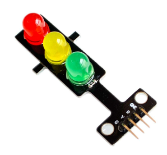
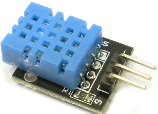
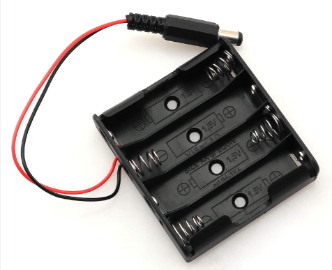
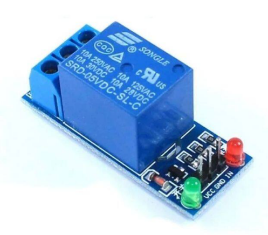
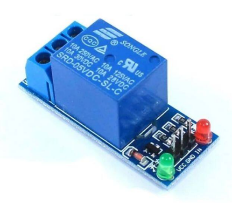
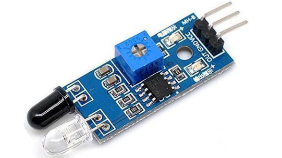
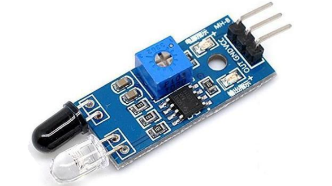
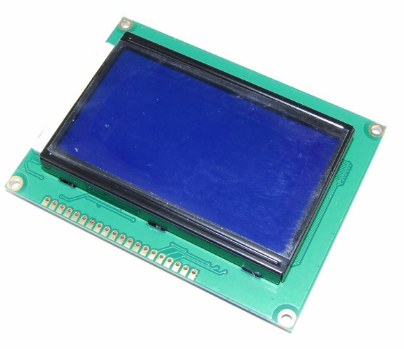
Ovaj dokument opisuje tehničke specifikacije i funkcionalnosti pametnog parking sistema zasnovanog na Arduino Mega mikrokontroleru, ESP8266 WiFi modulu, DFPlayer Mini MP3 modulu i DHT11 senzorima za monitoring temperature. Sistem omogućava rezervaciju parking mesta putem web stranice i automatski kontroliše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

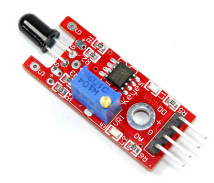


**2. Komponente Sistema**

* **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:**

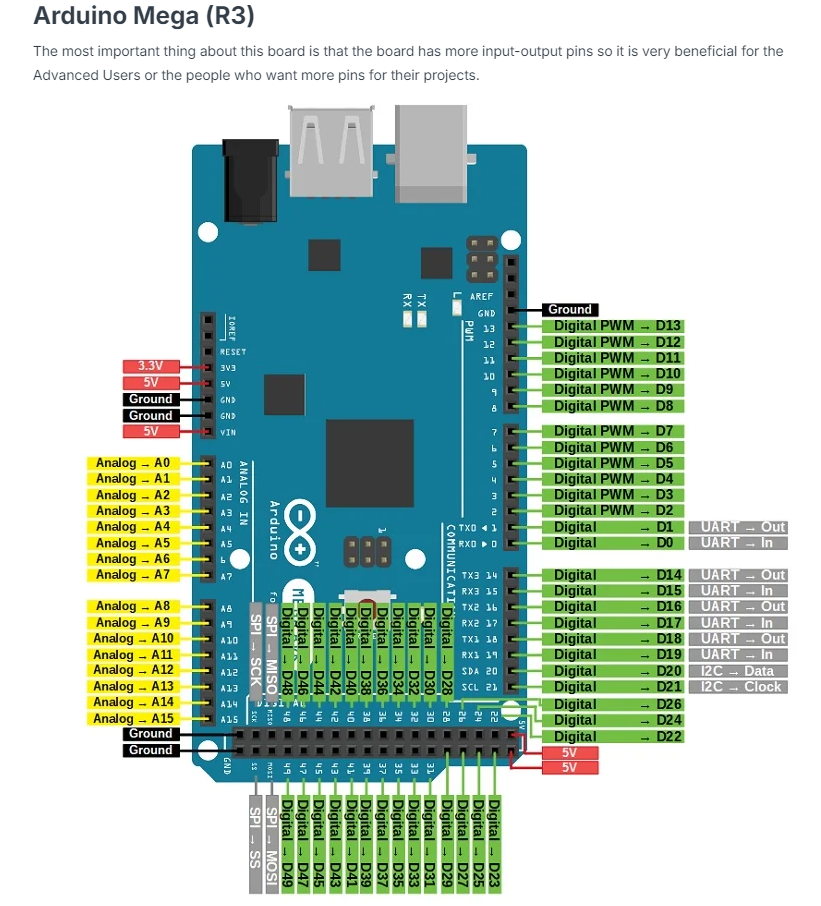
1. **Veliki broj pinova:** Sa 54 digitalna I/O pina i 16 analognih ulaza, Arduino Mega omogućava povezivanje sa više senzora i aktuatora istovremeno.
2. **Više serijskih portova:** Ploča poseduje 4 serijska porta (TX/RX parova) što omogućava komunikaciju sa više serijskih uređaja (npr. GPS, Wi-Fi moduli, 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**

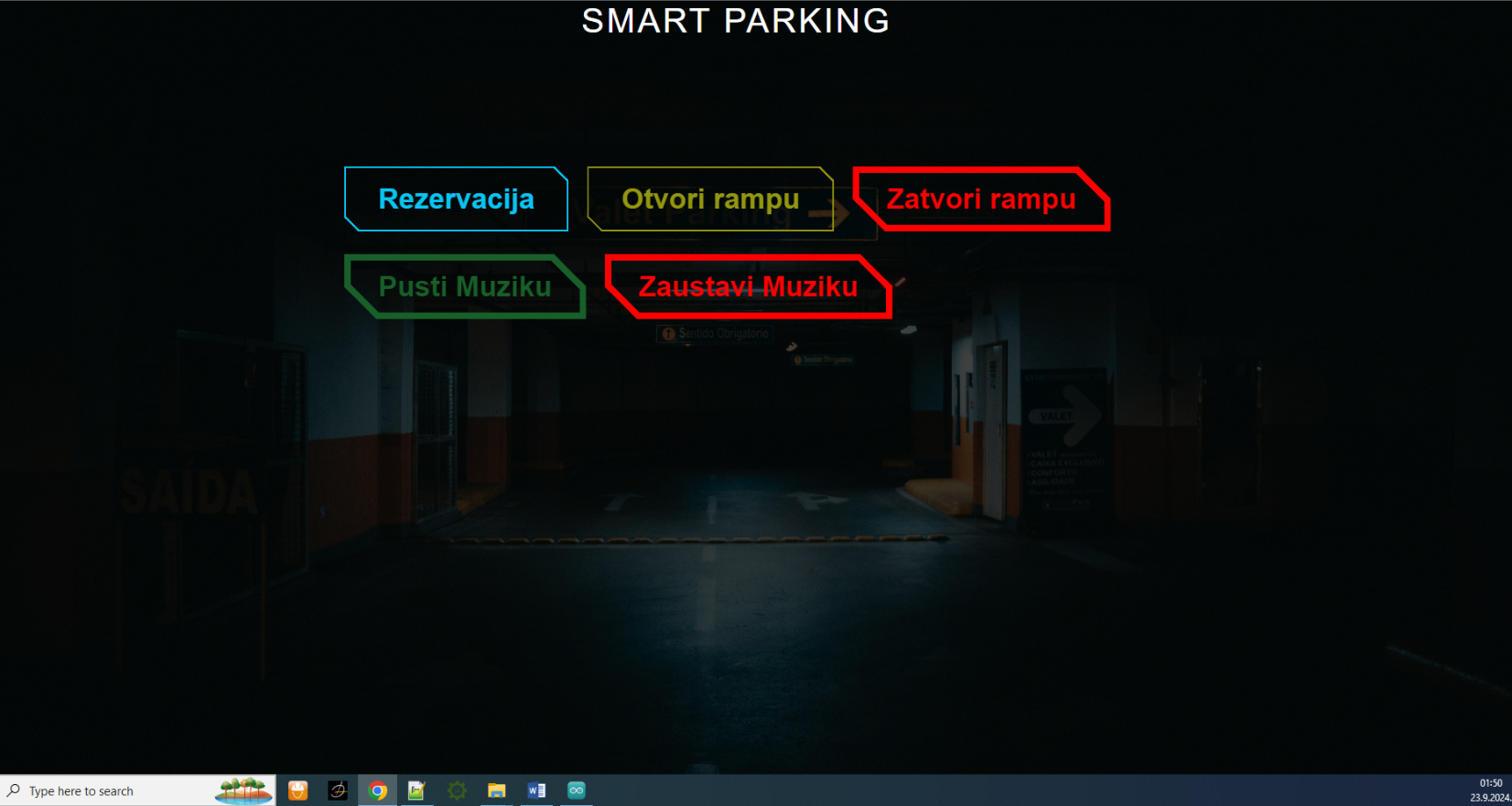
**4. Softverska Implementacija**

**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. #include <DHT11.h>
5. #include <SoftwareSerial.h> // SoftwareSerial library
6. #include <LiquidCrystal\_I2C.h>
7. #define servoPin 44
8. #include <U8g2lib.h>
9. #include <Servo.h>
11. #define trigPin\_3 43
12. #define echoPin\_3 42
14. Servo ServoRampa;
15. DFRobotDFPlayerMini myDFPlayer;
17. #define DHTPIN 42     // Pin na koji je povezan DHT senzor
18. #define DHTTYPE DHT11   // Tip senzora (DHT11 ili DHT22)
19. DHT dht(DHTPIN, DHTTYPE);  // Kreiramo instancu DHT senzora
20. int rxPin = 17; // RX2 pin na Arduino Mega
21. int txPin = 16; // TX2 pin na Arduino Mega
22. bool alarmAktiviran = false;
24. int IR\_Ulaz = 6;
25. int IR\_Izlaz = 7;
26. int ParkingMesta = 3;
27. int ZauzetaMesta = 0;
28. int Prvo\_Stanje = 0;
29. int Drugo\_Stanje = 0;
30. int piezoPin = 26;
31. long distance;
33. long distance\_2;
34. long distance\_3;
35. int buttonUlaz = 4;
36. int buttonState = 0;  // Trenutno stanje tastera
37. int lastButtonState = 0;  // Prethodno stanje tastera
38. bool isOpen = false;
39. bool carDetected = false;
40. bool carDetected2 = false;
41. const int AnalogWaterPin = A5;
42. const int SemaforCrvno\_1 = 22;
43. const int SemaforZuto\_1 = 23;
44. const int SemaforZeleno\_1 = 24;
45. const int SemaforCrvno\_2 = 38;
46. const int SemaforZuto\_2 = 39;
47. const int SemaforZeleno\_2 = 40;
48. const int SenzorPlamena = A0;
49. const int pragAktivacije = 60; // Prag za aktivaciju alarma
50. const int PumpPin = A2; // Pin na koji je povezan prekidač
51. const int relayPin = A4;
52. const int VentilatorPin = A8;
53. const int ReleyPinFan = A9;
54. bool espConnected = false;  // Promenljiva koja prati status konekcije ESP8266
56. enum StanjeSemafora {
57. CRVENO,
58. ZELENO,
59. BLINK\_ZELENO,
60. ZUTO\_1,
61. ZUTO\_2
62. };
63. StanjeSemafora trenutnoStanje = CRVENO;
64. StanjeSemafora trenutnoStanje2 = ZELENO;
65. unsigned long prethodniMillis = 0;
66. unsigned long prethodniMillis2 = 0;
67. const long trajanjeCrvenog = 7000;  // Trajanje crvenog svetla u milisekundama
68. const long trajanjeZelenog = 7000;  // Trajanje zelenog svetla u milisekundama
69. const long trajanjeBlinkanja = 4000; // Trajanje blinkanja zelenog svetla u milisekundama
70. const long trajanjeZutog = 2000;    // Trajanje žutog svetla u milisekundama
71. unsigned long blinkMillis = 0;  // Vreme za blinkanje zelenog svetla
72. bool zelenoBlinkStanje = false; // Trenutno stanje blinkanja zelenog svetla
73. //------------------------------------------------------------------------------------------------------
74. bool zelenoBlinkStanje2 = false; // Trenutno stanje blinkanja zelenog svetla drugog semafora
75. unsigned long blinkMillis2 = 0;  // Vreme za blinkanje zelenog svetla drugog semafora
76. // Definišemo minimalnu i maksimalnu udaljenost (u centimetrima)
77. const int minDistance = 5;    // Minimalna udaljenost (najbliže senzoru)
78. const int maxDistance = 10;  // Maksimalna udaljenost (najdalje od senzora)
79. // Ako koristite ST7920 sa SPI interfejsom
80. U8G2\_ST7920\_128X64\_F\_SW\_SPI u8g2(U8G2\_R0, /\* clock=\*/ 13, /\* data=\*/ 11, /\* CS=\*/ 10, /\* reset=\*/ 8);
81. void setup()
82. {
83. dht.begin();  // Inicijalizacija DHT senzora
84. Serial.begin(9600);    // Serijska komunikacija sa računarom
85. Serial2.begin(115200); // Serijska komunikacija sa ESP8266 (RX2/TX2)
86. Serial.println("Arduino Mega ready");
87. Serial2.println("Are you there, ESP8266?"); // Šalji proveru na ESP8266
88. pinMode(PumpPin, INPUT\_PULLUP);
89. pinMode(relayPin, OUTPUT);
90. digitalWrite(relayPin, LOW);       // Početno stanje releja je isključeno
92. pinMode(VentilatorPin,INPUT\_PULLUP);
93. pinMode(ReleyPinFan, OUTPUT);
94. digitalWrite(ReleyPinFan, LOW);
95. pinMode(buttonUlaz, INPUT\_PULLUP);
96. pinMode(31, OUTPUT);  // Trig pin kao izlaz
97. pinMode(30, INPUT);   // Echo pin kao ulaz
98. pinMode(50, OUTPUT);  // Trig pin kao izlaz
99. pinMode(51, INPUT);   // Echo pin kao ulaz
100. pinMode(45, OUTPUT);  // Trig pin kao izlaz
101. pinMode(44, INPUT);   // Echo pin kao ulaz
102. u8g2.begin(); // Inicijalizacija displeja
103. u8g2.setContrast(255); // Postavljanje maksimalnog kontrasta
104. pinMode(IR\_Ulaz, INPUT);
105. pinMode(IR\_Izlaz, INPUT);
107. pinMode(SemaforCrvno\_1,OUTPUT);
108. pinMode(SemaforZuto\_1,OUTPUT);
109. pinMode(SemaforZeleno\_1,OUTPUT);
110. digitalWrite(SemaforCrvno\_1, LOW);
111. digitalWrite(SemaforZuto\_1, LOW);
112. digitalWrite(SemaforZeleno\_1, LOW);
113. pinMode(SemaforCrvno\_2, OUTPUT);
114. pinMode(SemaforZuto\_2, OUTPUT);
115. pinMode(SemaforZeleno\_2, OUTPUT);
116. digitalWrite(SemaforCrvno\_2, LOW);
117. digitalWrite(SemaforZuto\_2, LOW);
118. digitalWrite(SemaforZeleno\_2, LOW);
119. pinMode(SenzorPlamena,INPUT);
121. trenutnoStanje = CRVENO;
122. prethodniMillis = millis();
123. trenutnoStanje2 = ZELENO;
124. prethodniMillis2 = millis();
126. ServoRampa.attach(servoPin);
127. ServoRampa.write(0);
129. }
130. void loop()
131. {
132. //-------------------------------------------------------------------------
133. Semafori();
134. ProveraKonekcije();
135. KontrolaVodenePumpe();
136. KontrolaVentilacije();
137. Vodostaj();
138. kontrolaUlazaIzlaza();
139. DHTSenzor();
140. proveriSenzorPlamena();
141. //----------------------------------NIVO VODE------------------------------------------------------------------
142. //-----------------------------------------PARKING SENZOR 1-----------------------------------------------------------
143. long distance = readUltrasonic(31, 30);
144. long distance\_2 = readUltrasonic(50, 51);
145. controlBuzzer(distance);
146. delay(100); // Mala pauza između merenja
148. controlBuzzer(distance\_2);  // Kontroliše pištanje za drugi senzor
149. delay(100); // Mala pauza između merenja
150. delay(100); // Kratka pauza pre sledećeg čitanja
151. if (distance < 5)
152. {
153. if (!carDetected)
154. {
155. carDetected = true;  // Označi da je auto detektovan
156. if (ParkingMesta > 0)
157. {
158. ParkingMesta--;  // Smanji broj slobodnih parking mesta
159. ZauzetaMesta++;
160. }
161. }
162. }
163. else
164. {
165. if (carDetected)
166. {
167. carDetected = false;  // Resetuj detekciju kada auto ode
168. ParkingMesta++;
169. ZauzetaMesta--;
170. }
171. }
173. if (distance\_2 < 5)
174. {
175. if (!carDetected2)
176. {
177. carDetected2 = true;  // Označi da je auto detektovan
178. if (ParkingMesta > 0)
179. {
180. ParkingMesta--;  // Smanji broj slobodnih parking mesta
181. ZauzetaMesta++;
182. }
183. }
184. }
185. else
186. {
187. if (carDetected2)
188. {
189. carDetected2 = false;  // Resetuj detekciju kada auto ode
190. ParkingMesta++;
191. ZauzetaMesta--;
192. }
193. }
194. //-----------------------------------------------
195. //---------------SPUŠTANJE RAMPE----------------------------
196. //-----------------------------------------------
197. if(Prvo\_Stanje==1 && Drugo\_Stanje==1)
198. {
199. ZatvoriRampu();
200. Prvo\_Stanje=0, Drugo\_Stanje=0;
201. }
202. //----------------------------------------------
203. u8g2.clearBuffer();          // Brisanje bafera
204. u8g2.setFont(u8g2\_font\_ncenB08\_tr); // Postavljanje fonta
205. u8g2.setCursor(0, 10);
206. u8g2.print("Slob. mesta: ");
207. u8g2.print(ParkingMesta);
208. u8g2.setCursor(0, 24);
209. u8g2.print("Zauz. mesta: ");
210. u8g2.print(ZauzetaMesta);
211. //------------------------PARKINZI NA DISPLEJU
212. u8g2.setCursor(0, 35);
213. u8g2.print("P1: ");
214. u8g2.print(distance);
216. u8g2.setCursor(55, 35);
217. u8g2.print("P2: ");
218. u8g2.print(distance\_2);
219. u8g2.setCursor(55, 45);
220. u8g2.print("P3: ");
221. u8g2.print(distance\_3);
222. //------------------------------------------------------------------------------------------------
223. //-----------------------------------------------------------------------------------------------
224. }
225. //----------------------------------------------------------------------VOIDI
226. void proveriSenzorPlamena() {
227. int stanjeVatre = analogRead(SenzorPlamena); // Čitanje vrednosti sa senzora plamena
228. Serial.println(stanjeVatre); // Ispisuje vrednost senzora na serijski monitor za debagovanje
229. if (stanjeVatre < pragAktivacije) { // Ako je vrednost ispod praga (požar detektovan)
230. tone(piezoPin, 1500);  // Aktivira pištanje piezo zvučnika
231. } else {
232. noTone(piezoPin);  // Isključuje pištanje piezo zvučnika
233. }
234. }
235. void DHTSenzor()
236. {
237. // Očitavanje temperature i vlažnosti
238. float vlaznost = dht.readHumidity();
239. float temperatura = dht.readTemperature();
240. // Provera da li su očitavanja uspešna
241. if (isnan(vlaznost) || isnan(temperatura)) {
242. Serial.println("Greska pri ocitavanju DHT senzora!");
243. return;
244. }
245. // Ispisivanje očitanih vrednosti na serijski monitor
246. Serial.print("Vlaznost: ");
247. Serial.print(vlaznost);
248. Serial.print(" %\t");
249. Serial.print("Temperatura: ");
250. Serial.print(temperatura);
251. Serial.println(" \*C");
252. delay(500);  // Pauza između očitavanja
253. }
254. void prikaziTekst(int tekstIndex)
255. {
256. u8g2.clearBuffer();
257. switch (tekstIndex) {
258. case 0:
259. u8g2.drawStr(0, 24, "Dobrodosli!"); // Prikazuje prvi tekst
260. break;
261. }
262. u8g2.sendBuffer(); // Slanje bafera na displej
263. }
264. unsigned long prethodnoVremeUlaz = 0;
265. const unsigned long vremeCekanja = 100; // Vreme čekanja od 100 milisekundi za debounce
266. void kontrolaUlazaIzlaza()
267. {
268. unsigned long trenutnoVreme = millis();
269. // Provera stanja tastera za otvaranje rampe
270. buttonState = digitalRead(buttonUlaz); // Čitanje trenutnog stanja tastera
271. if (buttonState != lastButtonState && (trenutnoVreme - prethodnoVremeUlaz >= vremeCekanja)) {
272. if (buttonState == LOW) {
273. OtvoriRampu();
274. isOpen = true;
275. }
276. isOpen = !isOpen; // Promena stanja rampe
277. prethodnoVremeUlaz = trenutnoVreme; // Ažuriraj vreme ulaza
278. }
280. // Provera ulaza na parking
281. if (digitalRead(IR\_Ulaz) == LOW && Prvo\_Stanje == 0)
282. {
283. Prvo\_Stanje = 1;
284. }
285. // Provera izlaza sa parkinga
286. if (digitalRead(IR\_Izlaz) == LOW && Drugo\_Stanje == 0 && (trenutnoVreme - prethodnoVremeUlaz >= vremeCekanja))
287. {
288. Drugo\_Stanje = 1;
289. if (Prvo\_Stanje == 0)
290. {
291. OtvoriRampu();
292. }
293. prethodnoVremeUlaz = trenutnoVreme; // Ažuriraj vreme izlaza
294. }
296. lastButtonState = buttonState; // Skladištenje trenutnog stanja tastera za sledeću iteraciju
297. }
298. void Vodostaj() {
299. int NivoVode = analogRead(AnalogWaterPin); // Čitanje nivoa vode sa analognog pina
300. u8g2.setCursor(0, 55); // Podešavanje kursora za prikaz na ekranu
301. u8g2.print("Voda:  ");
302. u8g2.print(NivoVode); // Prikazivanje nivoa vode
303. u8g2.sendBuffer();  // Slanje bafera na displej
304. delay(100); // Mala pauza pre sledećeg osvežavanja
305. }
306. void KontrolaVentilacije()
307. {
308. if(digitalRead(VentilatorPin) == HIGH)
309. {
310. digitalWrite(ReleyPinFan,HIGH);
311. }
312. else
313. {
314. digitalWrite(ReleyPinFan,LOW);
315. }
316. }
317. bool pumpState = false;  // Početno stanje pumpe - isključena
318. void KontrolaVodenePumpe() {
319. int StanjePumpe = digitalRead(PumpPin);
320. digitalWrite(relayPin, (StanjePumpe == LOW) ? HIGH : LOW);
321. }
322. void ProveraKonekcije()
323. {
324. float temperature = dht.readTemperature();
325. if (Serial.available())
326. {
327. String command = Serial.readStringUntil('\n'); // Čita komandu sa serijskog porta
328. command.trim(); // Uklanja eventualne razmake i prazne redove
329. if (command == "OPEN")
330. {
331. OtvoriRampu();
332. }
333. if (command == "CLOSE")
334. {
335. ZatvoriRampu();
336. }
337. if (command == "RESERVE1")
338. {
339. if (ParkingMesta > 0) {
340. ParkingMesta--;  // Smanjuje broj slobodnih parking mesta
341. ZauzetaMesta++;  // Povećava broj zauzetih parking mesta
342. Serial.println("Parking mesto je rezervisano.");
343. }
344. else
345. {
346. tone(piezoPin, 255);
347. }
348. }
349. if (command == "PLAY")
350. {
351. Serial.println("Pustanje prve pesme...");
352. myDFPlayer.setTimeOut(500);  // Serijski timeout 500ms
353. myDFPlayer.volume(25);       // Podešavanje jačine zvuka (opseg od 0 do 30)
354. myDFPlayer.EQ(0);            // Normalna ekvilizacija
355. myDFPlayer.play(1);  // Pusti prvu pesmu (0001.mp3) sa SD kartice
356. }
357. if(command == "STOP")
358. {
359. myDFPlayer.stop();
360. }
361. if (command == "GET\_STATUS") {
362. int SlobodnaMesta = ParkingMesta - ZauzetaMesta;
363. String statusMessage = String(SlobodnaMesta) + "," + String(ZauzetaMesta);
364. Serial.println(statusMessage);  // Šaljemo ESP8266 broj slobodnih i zauzetih mesta
365. Serial.println("Poslat status: " + statusMessage);  // Za debagovanje
366. }
367. }
368. }
369. void Semafori()
370. {
371. unsigned long trenutniMillis = millis();
372. unsigned long trenutniMillis2 = millis();
373. switch (trenutnoStanje)
374. {
375. case CRVENO:
376. digitalWrite(SemaforCrvno\_1, HIGH);
377. digitalWrite(SemaforZuto\_1, LOW);
378. digitalWrite(SemaforZeleno\_1, LOW);
379. if (trenutniMillis - prethodniMillis >= trajanjeCrvenog) {
380. trenutnoStanje = ZUTO\_1;
381. prethodniMillis = trenutniMillis;
382. }
383. break;
384. case ZUTO\_1:
385. digitalWrite(SemaforCrvno\_1, LOW);
386. digitalWrite(SemaforZuto\_1, HIGH);
387. digitalWrite(SemaforZeleno\_1, LOW);
388. if (trenutniMillis - prethodniMillis >= trajanjeZutog) {
389. trenutnoStanje = ZELENO;
390. prethodniMillis = trenutniMillis;
391. }
392. break;
393. case ZELENO:
394. digitalWrite(SemaforCrvno\_1, LOW);
395. digitalWrite(SemaforZuto\_1, LOW);
396. digitalWrite(SemaforZeleno\_1, HIGH);
397. if (trenutniMillis - prethodniMillis >= trajanjeZelenog) {
398. trenutnoStanje = BLINK\_ZELENO;
399. prethodniMillis = trenutniMillis;
400. blinkMillis = trenutniMillis;
401. }
402. break;
403. case BLINK\_ZELENO:
404. if (trenutniMillis - blinkMillis >= 500) {
405. blinkMillis = trenutniMillis;
406. zelenoBlinkStanje = !zelenoBlinkStanje;
407. digitalWrite(SemaforZeleno\_1, zelenoBlinkStanje ? HIGH : LOW);
408. }
409. if (trenutniMillis - prethodniMillis >= trajanjeBlinkanja) {
410. trenutnoStanje = ZUTO\_2;
411. prethodniMillis = trenutniMillis;
412. }
413. break;
414. case ZUTO\_2:
415. digitalWrite(SemaforCrvno\_1, LOW);
416. digitalWrite(SemaforZuto\_1, HIGH);
417. digitalWrite(SemaforZeleno\_1, LOW);
418. if (trenutniMillis - prethodniMillis >= trajanjeZutog) {
419. trenutnoStanje = CRVENO;
420. prethodniMillis = trenutniMillis;
421. }
422. break;
423. }
424. switch (trenutnoStanje2) {
425. case ZELENO:
426. digitalWrite(SemaforCrvno\_2, LOW);
427. digitalWrite(SemaforZuto\_2, LOW);
428. digitalWrite(SemaforZeleno\_2, HIGH);
429. if (trenutniMillis2 - prethodniMillis2 >= trajanjeZelenog) {
430. trenutnoStanje2 = BLINK\_ZELENO;
431. prethodniMillis2 = trenutniMillis2;
432. blinkMillis2 = trenutniMillis2;
433. }
434. break;
435. case BLINK\_ZELENO:
436. if (trenutniMillis2 - blinkMillis2 >= 500) {
437. blinkMillis2 = trenutniMillis2;
438. zelenoBlinkStanje2 = !zelenoBlinkStanje2;
439. digitalWrite(SemaforZeleno\_2, zelenoBlinkStanje2 ? HIGH : LOW);
440. }
441. if (trenutniMillis2 - prethodniMillis2 >= trajanjeBlinkanja) {
442. trenutnoStanje2 = ZUTO\_1;
443. prethodniMillis2 = trenutniMillis2;
444. }
445. break;
446. case ZUTO\_1:
447. digitalWrite(SemaforCrvno\_2, LOW);
448. digitalWrite(SemaforZuto\_2, HIGH);
449. digitalWrite(SemaforZeleno\_2, LOW);
450. if (trenutniMillis2 - prethodniMillis2 >= trajanjeZutog) {
451. trenutnoStanje2 = CRVENO;
452. prethodniMillis2 = trenutniMillis2;
453. }
454. break;
455. case CRVENO:
456. digitalWrite(SemaforCrvno\_2, HIGH);
457. digitalWrite(SemaforZuto\_2, LOW);
458. digitalWrite(SemaforZeleno\_2, LOW);
459. if (trenutniMillis2 - prethodniMillis2 >= trajanjeCrvenog) {
460. trenutnoStanje2 = ZUTO\_2;
461. prethodniMillis2 = trenutniMillis2;
462. }
463. break;
464. case ZUTO\_2:
465. digitalWrite(SemaforCrvno\_2, LOW);
466. digitalWrite(SemaforZuto\_2, HIGH);
467. digitalWrite(SemaforZeleno\_2, LOW);
468. if (trenutniMillis2 - prethodniMillis2 >= trajanjeZutog) {
469. trenutnoStanje2 = ZELENO;
470. trenutniMillis2 = trenutniMillis2;
471. }
472. break;
473. }
474. }
475. // Funkcija za spuštanje rampe
476. void OtvoriRampu()
477. {
478. slowMove(ServoRampa, 0, 90, 10);
479. }
480. void ZatvoriRampu()
481. {
482. slowMove(ServoRampa, 90, 0, 10);
483. }
484. void PunParking()
485. {
486. u8g2.setFont(u8g2\_font\_ncenB08\_tr); // Postavljanje fonta
487. u8g2.setCursor(0, 50);
488. u8g2.print("Parking je pun!");
489. u8g2.sendBuffer();           // Slanje bafera na displej
490. tone(piezoPin, 255);
491. delay(1000);  // Pauza od 1 sekunde
492. noTone(piezoPin);
493. }
494. //----------------------------------
495. //---------------------------------------
496. // Funkcija za sporo pomeranje servo motora
497. void slowMove(Servo &servo, int start, int end, int delayTime)
498. {
499. if (start < end) {
500. for (int pos = start; pos <= end; pos++) {
501. servo.write(pos);  // Postavljanje pozicije servo motora
502. delay(delayTime);  // Pauza za usporavanje kretanja
503. }
504. } else {
505. for (int pos = start; pos >= end; pos--) {
506. servo.write(pos);  // Postavljanje pozicije servo motora
507. delay(delayTime);  // Pauza za usporavanje kretanja
508. }
509. }
510. }
511. long readUltrasonic(int trigPin, int echoPin)
512. {
513. digitalWrite(trigPin, LOW);
514. delayMicroseconds(2);
515. digitalWrite(trigPin, HIGH);
516. delayMicroseconds(10);
517. digitalWrite(trigPin, LOW);
518. long duration = pulseIn(echoPin, HIGH);
519. long distance = (duration \* 0.034) / 2;
520. return distance;
521. }
522. void controlBuzzer(long distance) {
523. if (distance > 0 && distance < 5) { // Podesite maksimalnu udaljenost za detekciju
524. int frequency = map(distance, 0, 5, 2000, 100); // Invertovana logika: bliže objekat -> viša frekvencija
525. tone(piezoPin, frequency, 100); // Pišti 100 ms
526. } else {
527. noTone(piezoPin); // Zaustavi pištanje ako nema detekcije ili je objekat predaleko
528. }
529. }
530. //----------------------------------------------------------------------------------------------------------------
531. //----------------------------------------------------------------------------------------------------------------
532. void updateLCD() {
533. u8g2.clearBuffer();
534. u8g2.setFont(u8g2\_font\_ncenB08\_tr);
535. u8g2.setCursor(0, 10);
536. u8g2.print("Slob. mesta: ");
537. u8g2.print(ParkingMesta);
538. u8g2.setCursor(0, 24);
539. u8g2.print("Zauz. mesta: ");
540. u8g2.print(ZauzetaMesta);
541. u8g2.sendBuffer();
542. }
     * 1. **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)

