Parcare cu barieră pe bază de cartelă și senzori

Introducere

În cadrul acestui proiect se va concepe o parcare inteligentă, în care mașinile intră pe baza unei cartele. La apropierea cartelei, se va deschide bariera, iar mașina va fi contorizată ca fiind prezentă. La intrarea în parcare se vor afișa numărul de locuri libere, un loc fiind considerat liber atunci când senzorul din poziția respectivă nu detectează o mașină.

Descriere generală

La intrarea în parcare, va exista un display ce va afișa statistici legate de numărul de locuri libere din parcare. La apropierea cartelei se va ridica bariera și mașina va putea pătrunde în parcare unde își va putea alege un loc de parcare. Fiecare loc de parcare e dotat cu un senzor folosit, de asemenea pentru calculul numărului de locuri vacante.

Utilizatorii sunt identificați în funcție de cartelele folosite pentru intrare, respectiv părăsirea parcării, așadar se poate cunoaște mereu atât numărul de useri din parcare, cât și care sunt aceștia.

La părăsirea parcării se va contoriza numărul de locuri rămase libere, pentru a se putea permite o nouă intrare, ștergând din memorie cartela memorată la intrare.

Componente folosite

1. **Modul RFID-RC522**
   - Citește cartelele RFID pentru a permite accesul.
   - Interfață SPI
2. **Servo motor**
   - Ridică sau coboară bariera de acces.
   - Control prin semnal PWM
3. **Senzor ultrasonic HC-SR04**
   - Detectează prezența unei mașini în zona barierei.
4. **Afișaj LCD**
   - Afișează mesaje și numărul de locuri disponibile.
   - Interfață I2C
5. **LED-uri (roșu/verde)**

- Indică vizual dacă un loc este liber sau nu.

Hardware Design

Listă de piese și pini

Servo motor

Pin componentă	Pin Arduino
Semnal	D3
GND	GND
VCC	VCC

Senzor Ultrasonic HC-SR04

Pin componentă	Pin Arduino
Echo	D6
Trigger	D5
GND	GND
VCC	VCC

LED-uri

- LED roșu → Pin D7
- LED verde → Pin D2

RFID-RC522

Pin componentă	Pin Arduino
SDA	D10
SCK	D13
MOSI	D11
MISO	D12
RST	D9
GND	GND
3V3	3.3

I2C Liquid Crystal Display

Pin componentă	Pin Arduino
SCL	A5
SDA	A4
GND	GND
VCC	VCC

Software Design

Mediu de dezvoltare

Pentru dezvoltarea proiectului s-a folosit Arduino IDE.

Librarii si surse third-party

`Servo.h` - ridicarea barierei la intrare și ieșire

`LiquidCrystal_I2C.h` - afișarea de mesaje corespunzătoare, cât și a numărului de locuri libere

`MFRC522.h` - utilizarea modulului RFID pentru acționarea barierei pe bază de cartelă

`SPI.h` - modulul RFID folosești comunicare SPI, astfel avem nevoie de biblioteca respectivă

Algoritmi și structuri de date

Logarea unui user: atunci când se scanează o cartelă, se va identifica pe baza unui **vector de șiruri de caractere** dacă acel user se află în parcare sau nu. Pe baza acestui rezultat se va afișa un mesaj de tipul “Hello!” sau “Goodbye!”

Eliminare user: vom stoca dinamic userii logați, astfel, atunci când un user va părăsi parcarea, pentru a nu irosi memorie, vom rearanja userii în memorie pentru a stoca eficient într-o zonă contiguă de memorie

Identificare loc liber: în momentul în care mașina parchează, vom identifica acest lucru și vom schimba becul LED ce se află aprins. Pentru a identifica dacă există sau nu o mașină, s-au folosit informațiile primite de la senzorul de distanță, un prag limită find setat pentru 5cm.

Afișare locuri libere: în permanență, pe ecranul parcării se va afișa numărul de locuri rămase libere în parcare. Acest lucru se calculează dinamic, actualizându-se numărul ori de câte ori un user intră sau părăsește parcarea.

Surse şi funcţii implementate

Github: https://github.com/rafaelchitan/SmartParking.git

Funcțiile necesare pentru buna desfășurare a logicii programului au fost împărțite după cum urmează:

- **`getDistance()`** - calculează distanța de la senzorul de parcare pentru a identifica dacă se află sau nu o mașină parcată acolo
- **`lightSpot()`** - în funcție de distanța până la senzor, se aprinde LED-ul corespunzător(roșu - există mașină parcată sau verde - locul este liber)

```
  void lightSpot(float distance) {
      if (distance < 5) {
        digitalWrite(GREEN_PIN, LOW);
        digitalWrite(RED_PIN, HIGH);
      } else {
        digitalWrite(GREEN_PIN, HIGH);
        digitalWrite(RED_PIN, LOW);
      }
  }
```

- **`storeUIDToCharArray()`** - citim UID-ul cartelei RFID de la receiver, pe care îl stocăm într-un array de caractere pentru a-l manipula mai ușor în program

```
void storeUIDToCharArray(char uidStr[]) {
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    char hexByte[4];
    sprintf(hexByte, "%02X", mfrc522.uid.uidByte[i]);
    strcat(uidStr, hexByte);
  }

  Serial.print("UID as char array: ");
  Serial.println(uidStr);
}
```

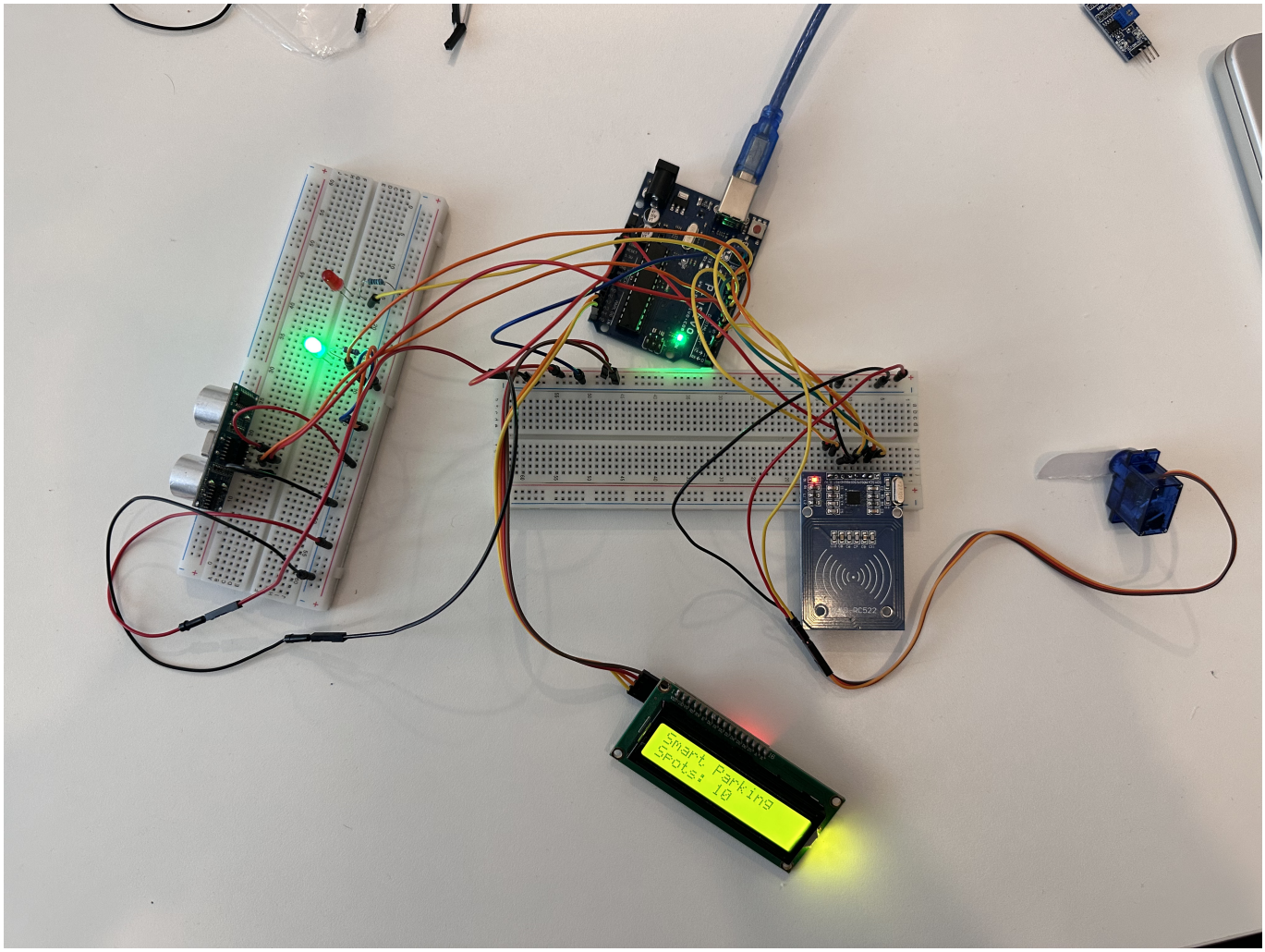
- **`openBarrier()`** - deschidem bariera ce permite trecerea, iar după o perioadă de timp, se revine la poziția inițială
- **`logUser()`** - în funcție de UID-ul cartele, identificăm dacă user-ul este în parcare sau nu și menținem această informație într-un array

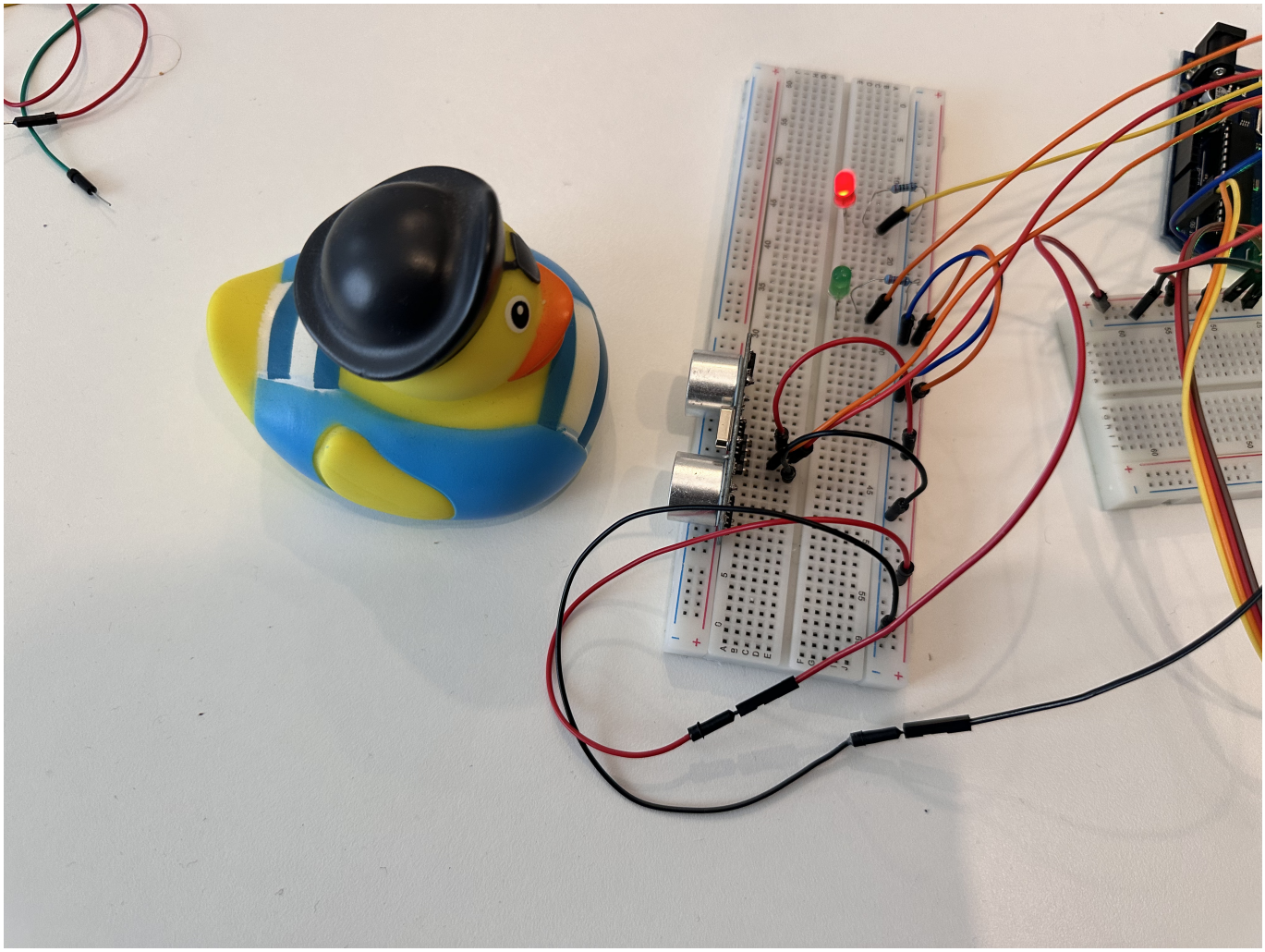
```
bool logUser(char uid[]) {
  for (int i = 0; i < numberOfUsers; i++)
    if (!strcmp(users[i], uid)) {
      Serial.println("User leaves");
      for (int j = i + 1; j < numberOfUsers; j++)
        strcpy(users[i], users[j]);
      strcpy(users[numberOfUsers - 1], "");
      numberOfUsers--;
      return false;
    }
```

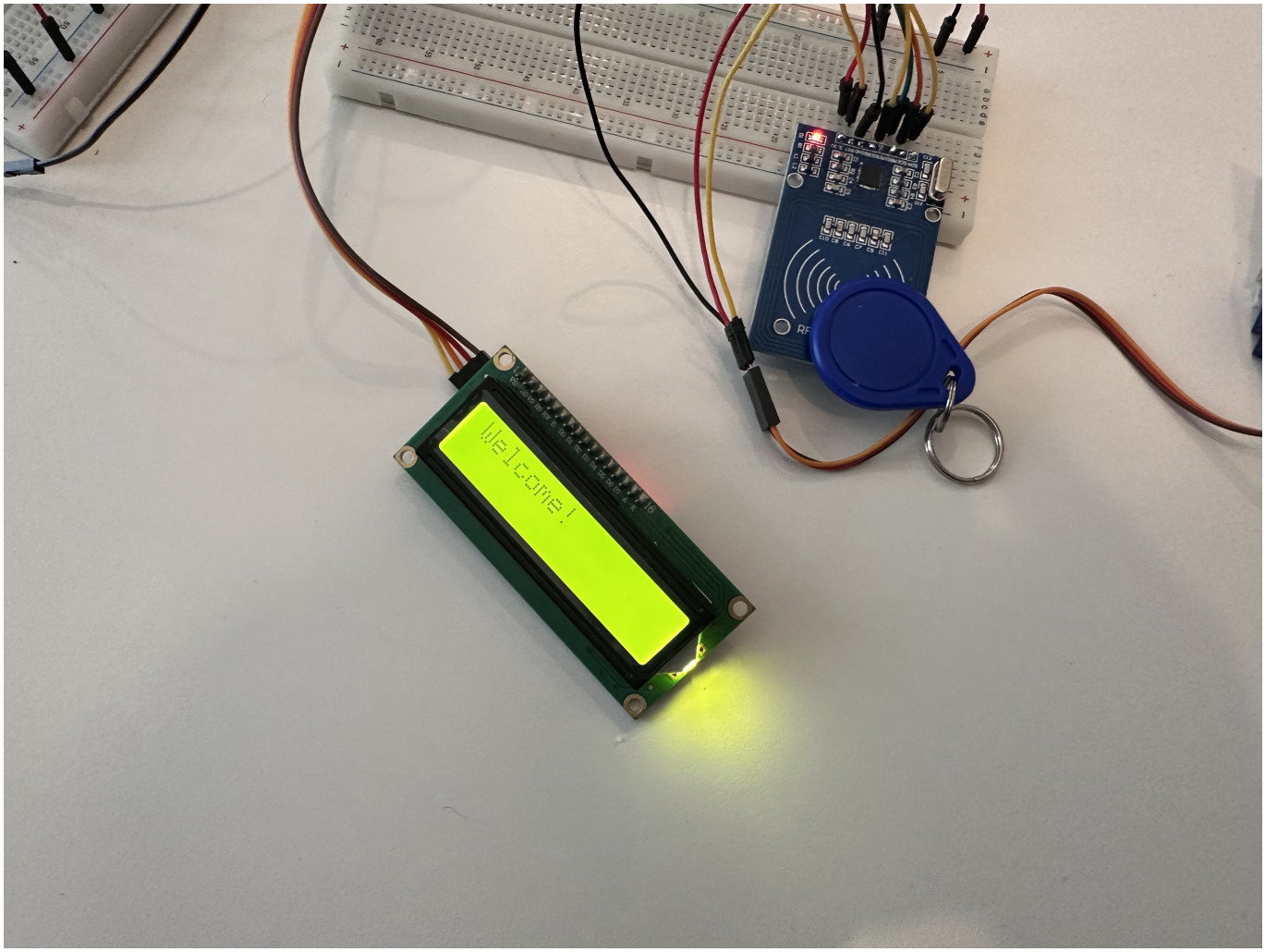
```
    strcpy(users[numberOfUsers++], uid);
    Serial.println("New user entered");
    return true;
}
```

- **`printMessage()`** - în funcție de user-ul ce scanează cartela, vom identifica dacă vrea să părăsească parcarea, sau să intre, pentru a afișa un mesaj corespunzător

Rezultate Obținute







Bibliografie/Resurse

Export to PDF