# Documentație

Simularea unei mașini [Parcare]

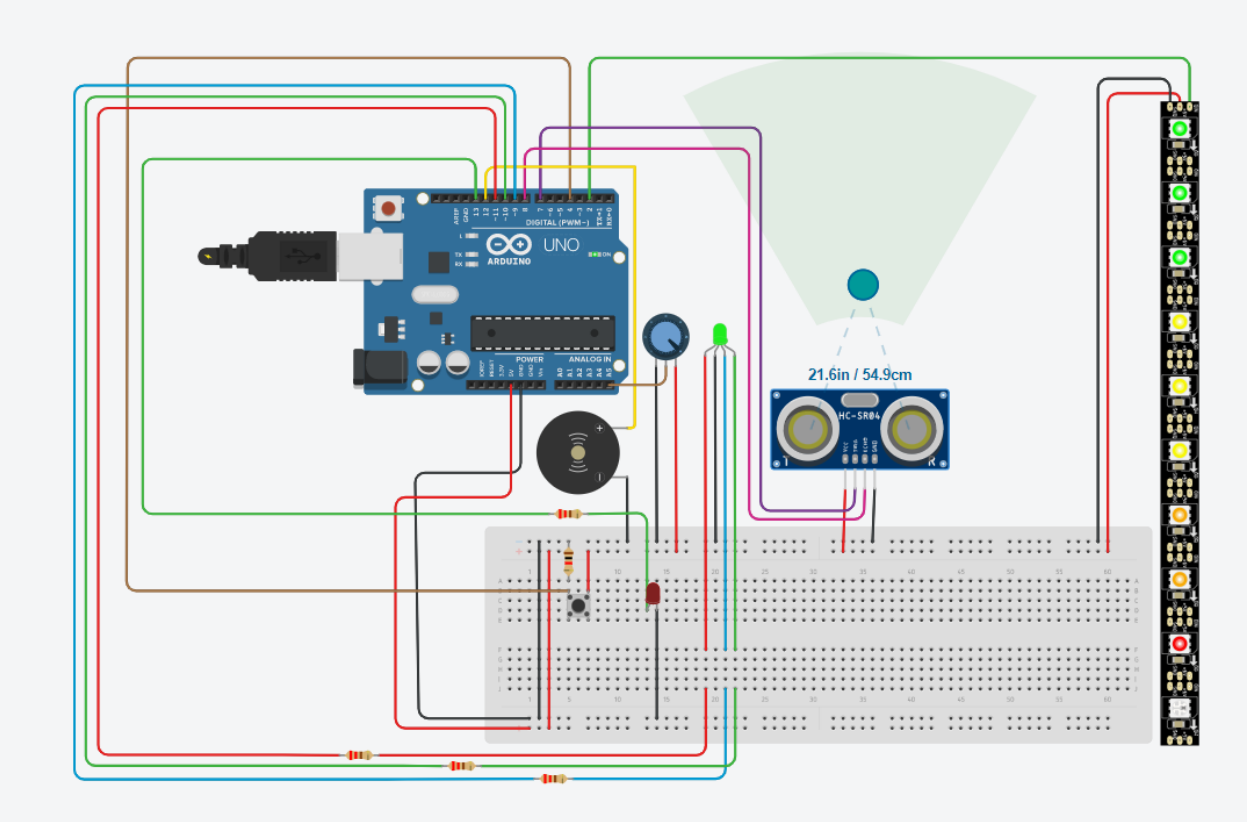
**Link Tinkercad**

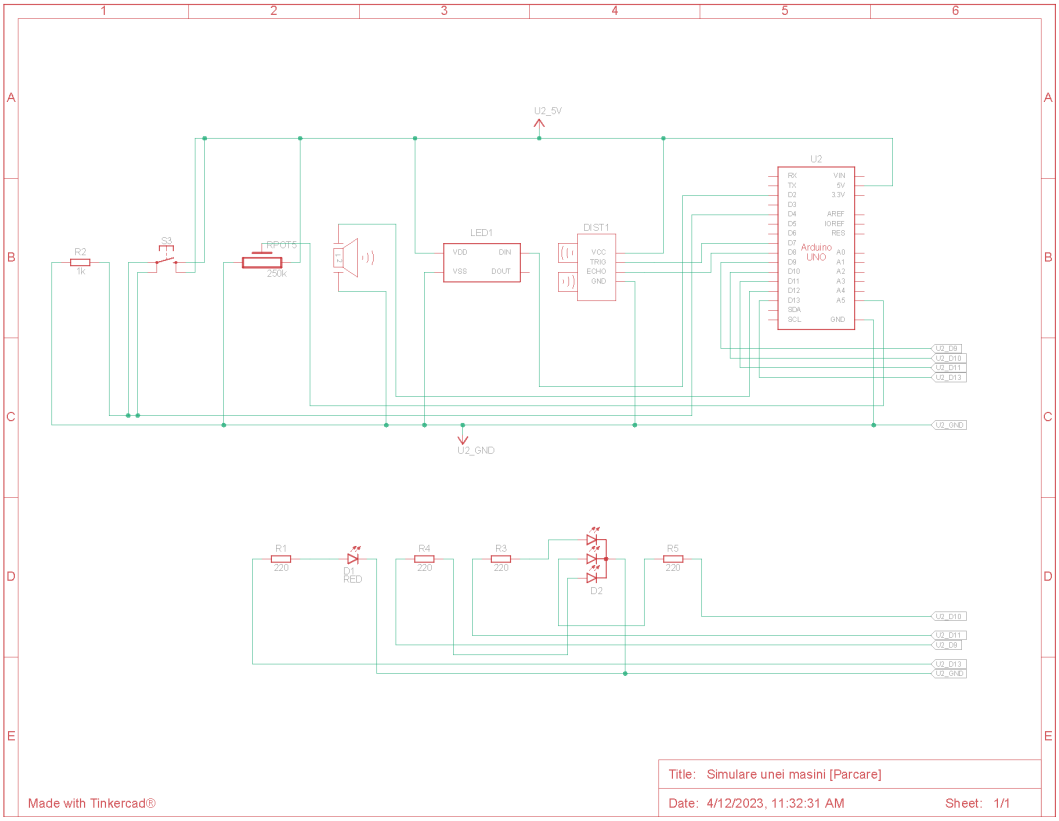
https://www.tinkercad.com/things/kArh4mG4iPT?sharecode=q1m2EMEptDSlAtCl8Rfe45BeiUjoeNZihkr8LnvWs4c

**Componente**

| NUME | PINI | INFO | CONECTAT LA PINII ARDUINO | TESTE |
|---|---|---|---|---|
| LED | Catod, Anod | Folosește o rezistență de 220Ω | GND, 13 digital | După cod predefinit |
| LED RGB | Roșu, Catod, Albastru, Verde | Folosește cate o rezistență de 220 Ω pentru fiecare culoare | 11 digital, GND, 9 digital, 10 digital | După cod predefinit |
| Neopixel Strip | GND, 5V, DIN | | GND, 5V, digital 2 | După cod predefinit |
| Arduino | - | | - | |
| Potențiometru | Terminal 1, Wiper, Terminal 2 | | GND, 5V, analog 5 | După cod predefinit |
| Senzor Ultrasonic | Vcc, Trig, Ech, GND | | 5V, 7 digital, 8 digital, GND, | După cod predefinit, debug folosind Serial.print pe distanța detectată |
| Buzzer | Pozitiv, Negativ | | Digital 12, GND | După cod predefinit |
| Buton | Catod, Anod | Folosește o rezistenta de 10KΩ | 5V, 4 digital | După cod predefinit, debug folosind Serial.print pe starea butonului |

## Arhitectura Hardware



## Arhitectura Software

Cod: Verificăm starea motorului. Dacă nu este oprit, putem porni senzorul de parcare. Acesta va aprinde LED-urile bării de stare corespunzător distanței dintre mașină și obiectul identificat. De asemenea va activa un semnal sonor conform ultimii culori activate. Dacă se oprește motorul, senzorul va fi dezactivat.

## Funcții

Definite de noi:

- **int getBarState(int reading)** - determină dacă butonul a fost apăsat
- **int getFinalPixel(int inches)** - determină ultimul pixel ce ar trebui aprins conform distanței dintre mașină și obiect
- **int getSoundDuration(int pixel)** - determină frecvența sunetului făcut de buzzer la un anumit nivel
- **void splitDelay(int duration)** - împarte parametrul primit în delay-uri de 10 ms.
- **int playSound(int inches)** - activează buzzer-ul la frecvența necesară
- **void choosePixelColor(int i)** - determină culoarea pixelului
- **void setBarPixels(int inches)** - colorează toți pixelii bării de stare conform distanței dintre mașină și obiect
- **void resetPixels(int start, int end)** - resetează pixelii dintr-un interval dat
- **bool setLedStatus(int value)** - determina dacă mașina este oprită, în accesorii sau pornită și returnează true dacă nu este oprită
- **void parkingSensor(bool isRunning, int barState)** – conține toate condițiile și instrucțiunile senzorului de parcare, pornind de la parametrii de intrare: starea motorului/contactului și starea butonului pentru oprire/pornire funcționalitate

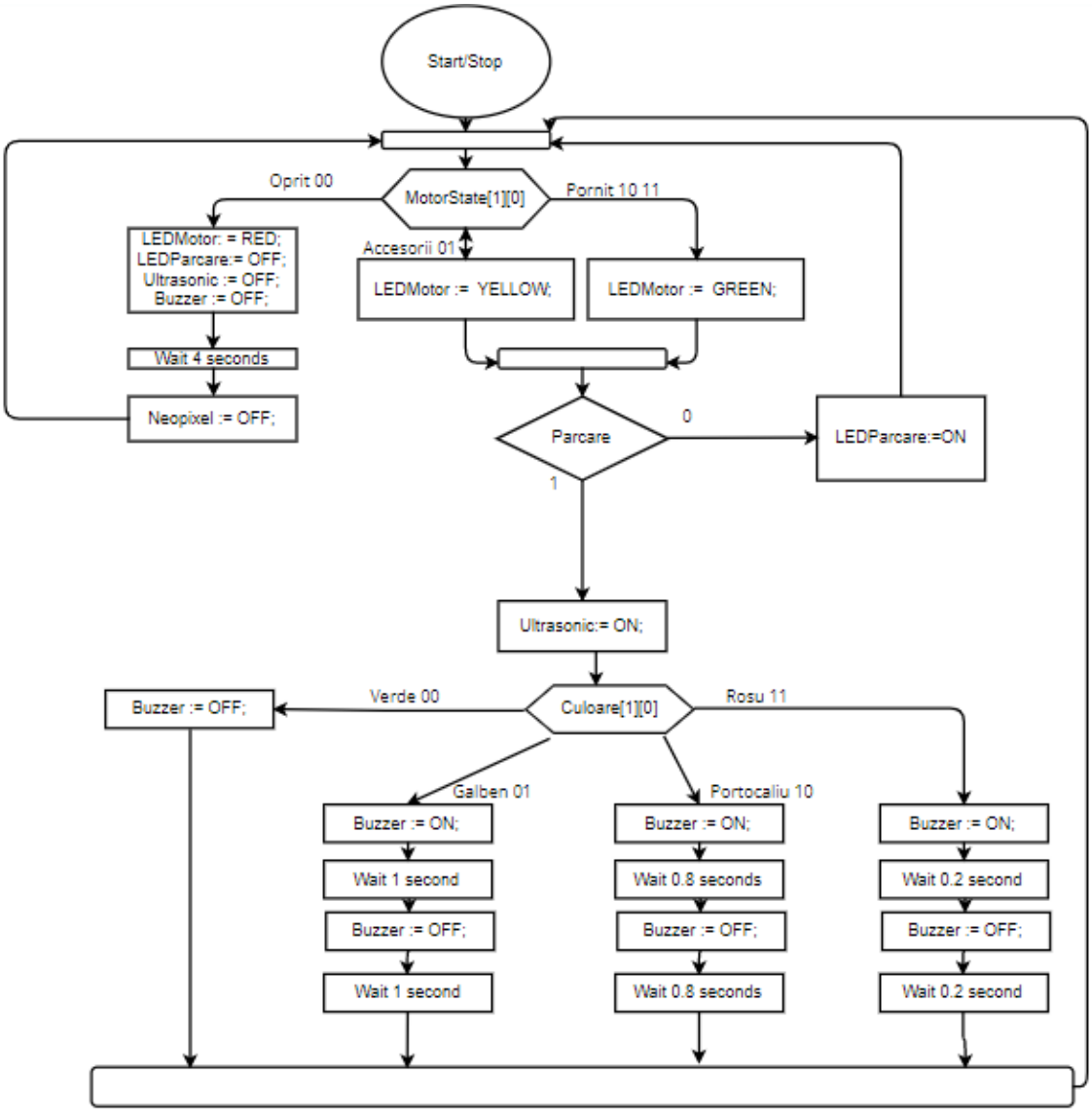
Predefinite:

- **long readUltrasonicDistance(int triggerPin, int echoPin)** - află distanța dintre mașină și obiect
- **void setup()** - inițializează librăria Neopixel și pinii de intrare și ieșire
- **void loop()** - repetă constant instrucțiunile din interiorul său
- **tone(BUZZER_PIN, BUZZER_NOTE, duration)** - alege tonul sunetului produs de buzzer
- **delay(duration)** - amână instrucțiunile pentru o durată dată
- **noTone (BUZZER_PIN)** - buzzer-ul nu va produce sunet
- **pixels.setPixelColor(i, pixels.Color(COLOR))** - setează pixelul ***i*** al unei bări pe culoarea dată
- **pixels.show()** - activează pixelii
- **digitalWrite(triggerPin, LOW/HIGH)** - activează sau dezactivează PIN-ul dat
- **digitalRead(BUTTON_PIN)** - citește starea unui PIN digital
- **analogRead(BUTTON_PIN)** - citește starea unui PIN analog
- **pinMode(BUTTON_PIN, INPUT/OUTPUT)** - alege modul pinului
- **millis()** - returnează de cât timp este activ sistemul

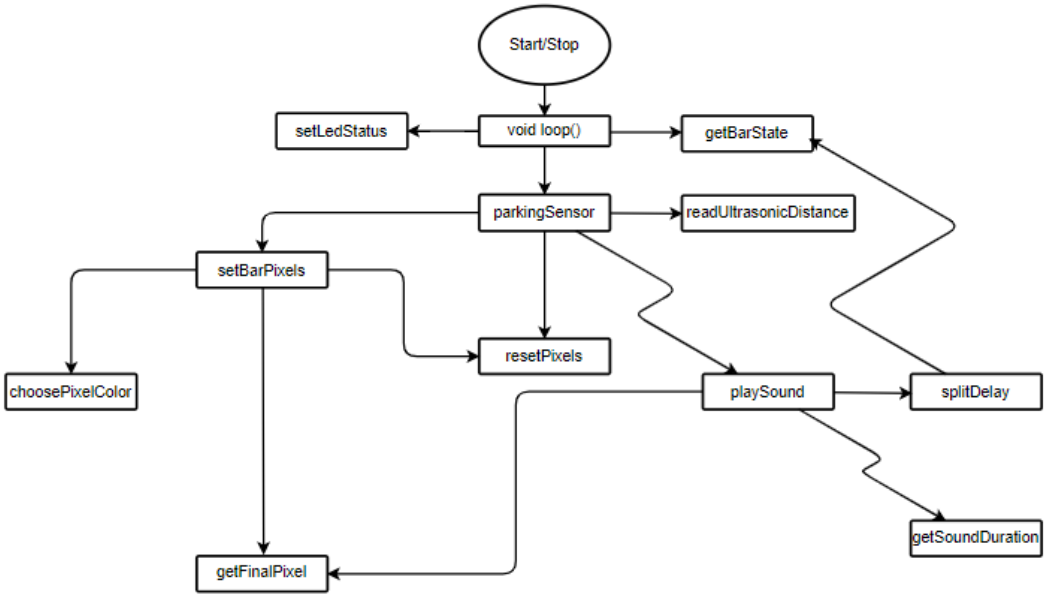
Librării:

- Adafruit_NeoPixel.h, folosită pentru bara ce arată distanța față de un obiect

## Schemă Logică



## Schema apelării funcțiilor




## Concluzii

Dificultăți:

- folosirea debouncerului pentru butonul ce activează parcarea
- Conectarea potențiometrului
- Conectarea LED-ului martor

Optimizări:

- Factorizarea aprinderii ledurilor de pe bara de stare

## Bibliografie


- https://docs.arduino.cc/built-in-examples/digital/Debounce