Automatic parking gate

Nica Mioara Raluca - 334CA

Introduction

Project Summary

The Automatic Parking Gate project simulates a real-world automated parking system with controlled access. It allows vehicles to enter only if they have valid authorization (via an RFID card) and only if parking spots are available. The system uses a servo motor to raise and lower a barrier, ultrasonic sensors to detect vehicle presence, an LCD to display the number of available spots, and colored LEDs to indicate the status of the parking lot.

The purpose of this project is to develop a smart access control system for parking areas, using multiple electronic modules and key concepts learned during laboratory sessions—such as RFID communication, PWM control, sensors, LCD interfacing, and interrupt handling. The goal is to build a functional and interactive prototype that simulates a secure and efficient real-life parking gate.

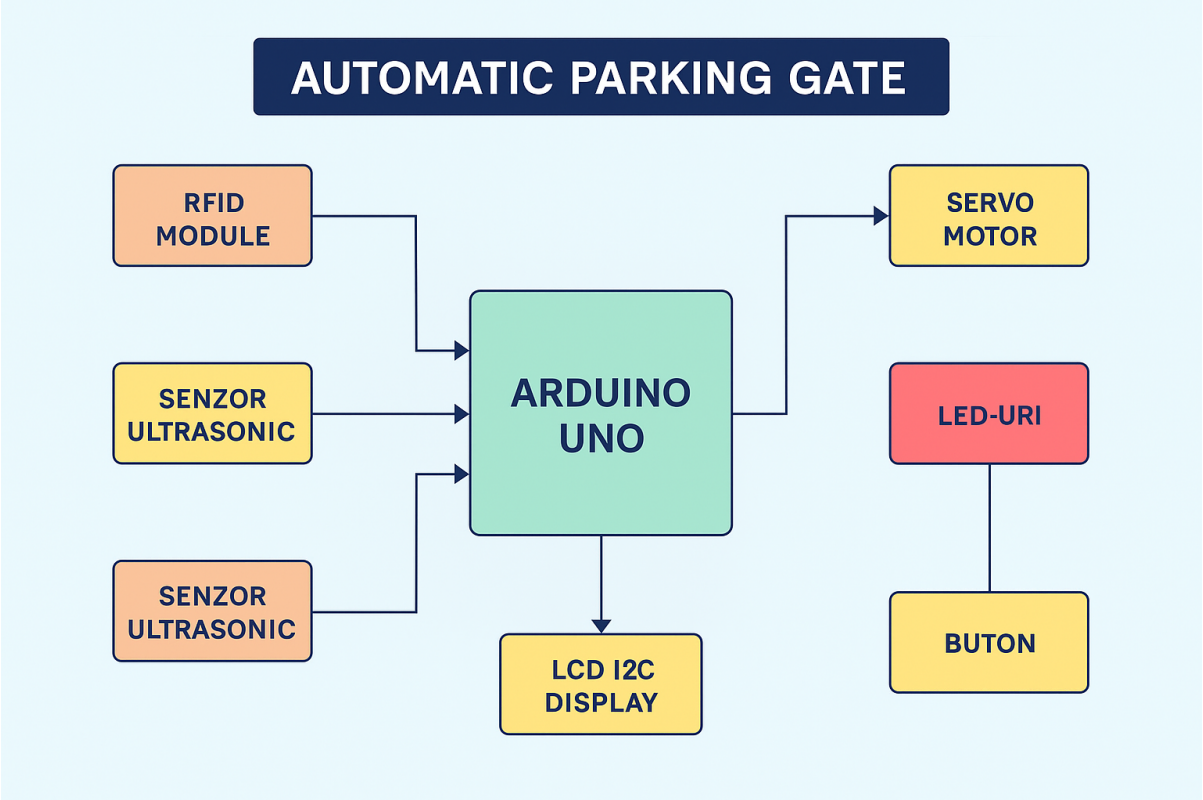
The idea for the project came from the observation that managing parking lots in crowded urban areas is increasingly difficult without automation. Many institutions and residential areas already use automated gates with card access and visual status indicators. This project aims to replicate such a system at a smaller scale, making use of accessible hardware and the skills acquired in class.

This project is **useful** as it demonstrates a practical implementation of several microcontroller-based technologies in a single integrated system. For us, it provided valuable experience in system design, hardware-software integration, and real-world problem solving. For others, it can serve as a learning resource or as a base for further development of smart parking or access control systems. It is educational, expandable, and applicable to real-life needs.

General Description

The Automatic Parking Gate project is a smart system that manages the entry of vehicles into a parking area based on access authorization (via RFID) and space availability (detected by ultrasonic sensors). The system integrates both hardware and software components that interact with each other in a coordinated manner, managed by the Arduino UNO microcontroller.

Below is the block diagram of the system, showing all modules involved:



Hardware Modules

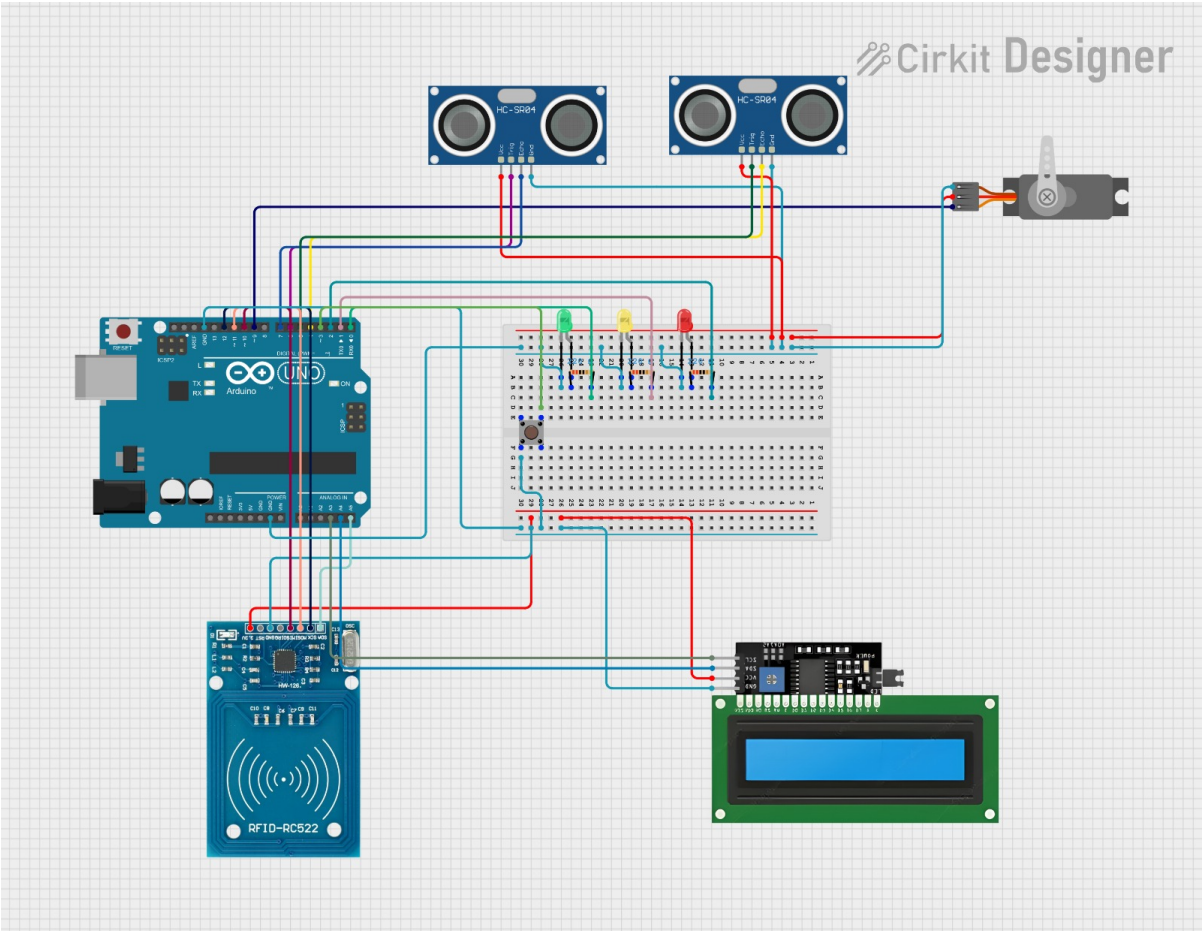
- **Arduino UNO** Acts as the central controller, processing data from sensors and sending commands to output devices.
- **RFID Module** (RC522) Reads the UID of the scanned card via SPI. The Arduino compares this UID to a list of authorized IDs.
- **Ultrasonic Sensors** (x2) Detect vehicle presence at the entrance (sensor 1) and exit (sensor 2). This helps update the count of available parking spots.
- **Servo Motor** Controls the parking barrier, raising it if access is granted and lowering it after the car has passed.
- **LCD 16×2 with I2C Module** Displays dynamic information such as the number of available spots or access denied messages.
- **LEDs** (green, orange, red) Provide quick visual feedback:
  1. Green → available
  2. Orange → almost full
  3. Red → full

Software Flow & Interactions

- The RFID module reads the card → Arduino checks if UID is authorized.
- If valid, ultrasonic sensor 1 checks if a car is present.
- If a vehicle is detected and parking space is available:
  - Servo motor raises the barrier.
  - LCD shows: “Access Granted | Free spots: X”.

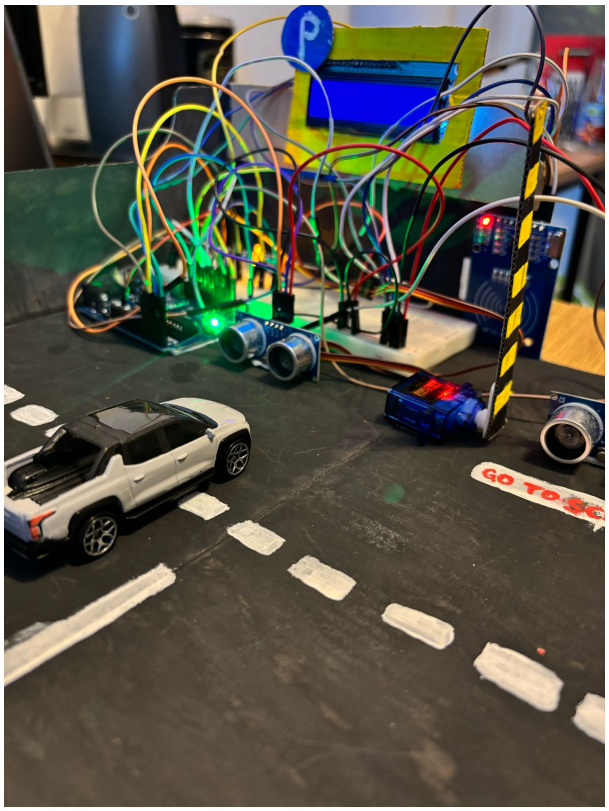
- Green LED turns on.
- If the parking is full or card is unauthorized:
  - Barrier stays closed.
  - LCD displays appropriate message.
  - Red LED turns on.
- After car enters, sensor 2 updates spot count.

Hardware Design



Component	Quantity	Arduino Pin Connections	Description / Role
Arduino Uno	1	–	Microcontroller central unit
RFID-RC522	1	VCC→3.3 V, GND→GND, SDA(SS)→D10, SCK→D13, MOSI→D11, MISO→D12, RST→D8	Reads card UID via SPI
HC-SR04 Ultrasonic Sensor	2	**Sensor 1:** VCC→5 V, GND→GND, Trig→D2, Echo→D4 **Sensor 2:** VCC→5 V, GND→GND, Trig→D5, Echo→D6	Detects vehicle presence (entrance/exit)
SG90 Servo Motor	1	VCC→5 V, GND→GND, Signal→D9	Raises/lowers the parking barrier
16×2 LCD w/ I²C Backpack	1	VCC→5 V, GND→GND, SDA→A4, SCL→A5	Displays spot count and status messages
Green LED	1	Anode→D7 (via 220 Ω), Cathode→GND	“Available spots” indicator

Orange LED	1	Anode→A0 (via 220 Ω), Cathode→GND	“Almost full” indicator
Red LED	1	Anode→A1 (via 220 Ω), Cathode→GND	“Full” indicator
220 Ω Resistors	3	In series with each LED	Current limiting for LEDs
Breadboard	1	–	Prototyping / common power rails
Jumper Wires	~20	–	Signal and power connections



Software Design

Development Environment

- **IDE:** PlatformIO in VS Code
- **Tool-chain:** avr-gcc + avrdude (bundled with the IDE)
- **MCU clock:** 16 MHz
- **Debug:** Serial Monitor @ 9600 baud; optional logic-analyzer capture

Libraries & Sources

- Wire – I2C bus (Arduino core)
- LiquidCrystal_PCF8574 – 16×2 LCD via PCF8574 backpack

- SPI – hardware SPI for MFRC522 (Arduino core)
- MFRC522 – 13.56 MHz RFID reader API (miguelbalboa/MFRC522)
- Servo – PWM control for the barrier arm
- avr/io.h – direct register access for fast LED updates

Architecture & Algorithms

State Flow

```
  1. Idle / Scan Card — LEDs show occupancy, LCD “Scan card…”.
  2. Entry Granted — valid UID and free spots.
  3. Raise Barrier (Entry) — ultrasonic #1 detects a vehicle and ≥ 10 s
since last raise.
  4. Count Entry — ultrasonic #2 sees vehicle, barrier lowers, carsInside++.
  5. Raise Barrier (Exit) — carsInside > 0, ultrasonic #2 sees vehicle and
cooldown met.
  6. Count Exit — vehicle clears sensor, barrier lowers, carsInside--.
```

Key Data

```
  ```cpp
  const byte authorizedUID[][4];   // whitelist of UIDs
  int  carsInside;                 // vehicles inside
  unsigned long lastRaiseMillis;   // last time barrier was raised
  ```
```

Timing Constants

- minRaiseInterval = 10 000 ms – global cooldown between raises
- carClearTime = 1 500 ms – confirm sensor field is empty before lowering

Source Files & Functions

- **parking_barrier.ino**
  - setup() – hardware init, “Welcome” splash
  - loop() – full FSM for entry/exit
  - readDistanceCm() – HC-SR04 helper
  - isAuthorized() – UID whitelist check
  - updateLEDs() – green/orange/red status LEDs
  - showMessage() – text output on LCD

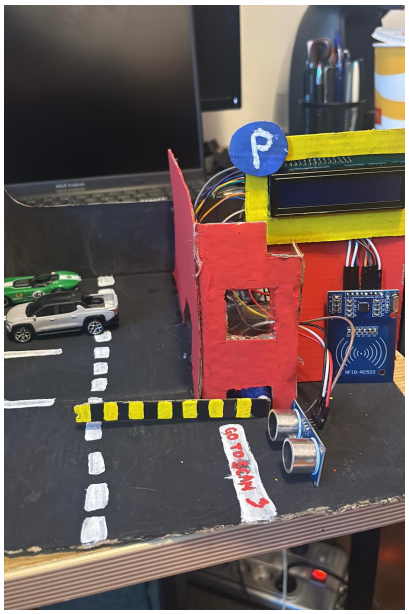
Planned Enhancements

- Safety photocell to block lowering if an obstacle is detected
- EEPROM-stored whitelist with master-card admin functions
- Web dashboard (ESP8266/ESP32) for live telemetry and OTA updates

Rezultate Obţinute

Code: https://github.com/Raluca1304/Automatic_parking_gate.git

Demo: https://youtube.com/shorts/BB8ATAf_ZLA?feature=share



Conclusions

The **Automatic Parking Gate** prototype successfully met its main goals: RFID-based access control, real-time spot counting with ultrasonic sensors, and safe barrier actuation via a servo—all accompanied by clear LCD messages and status LEDs. Bench-top and live tests showed:

* **Reliability** – The barrier opened only for authorised cards and never allowed occupancy to exceed capacity in all test scenarios.

* **Response time** – The arm rises in under 1 s after a valid scan and lowers about 800 ms after the vehicle clears the exit sensor.

* **User feedback** – LCD prompts and the green / yellow / red LED scheme gave immediate, intuitive information.

On the learning side, the project reinforced skills in:

* SPI and I2C communication on Arduino hardware;

* designing a clean finite-state machine;

* integrating multiple hardware modules into a cohesive system.

Observed limitations * Ultrasonic sensors can mis-trigger in heavy rain or on very angled surfaces.

* The whitelist of cards is hard-coded—any change requires reflashing.

* There is no dedicated safety sensor to halt the arm while it is lowering.

Future improvements

1. Add an IR **safety photocell** to stop the barrier if something crosses underneath.

2. Store the whitelist in **EEPROM** and use a master card for live admin tasks.

3. Attach a Wi-Fi module (ESP8266/ESP32) for a web dashboard, live telemetry and OTA firmware updates.

4. Enclose the electronics in a 3-D-printed, weather-proof housing and provide rugged power filtering.

Implementing these upgrades would move the prototype closer to a deployable, real-world parking solution—enhancing safety, flexibility and ease of maintenance.

Download

automatic_parking_gate-master.zip

Bibliography / Resources

Software Resources

* **Arduino Core Documentation** – “Language Reference” and “Core Libraries” (arduino.cc)

* **MFRC522 Library** – GitHub repository *miguelbalboa/MFRC522*, release v1.4.11 README and example sketches for SPI RFID communication

* **LiquidCrystal_PCF8574 Library** – GitHub repository *mathertel/LiquidCrystal_PCF8574*, release v1.3.x I2C backpack usage and custom character guide

* **PlatformIO Documentation** – “Getting Started with AVR/Arduino” and “Library Dependency Finder”

* **AVR Libc Manual** – chapter on direct register access (`avr/io.h`)

* **Timer-1 Fast-PWM Tutorial** – Nick Gammon’s blog post on generating 50 Hz servo signals

* **Microchip ATmega328P Datasheet** – section on timers/counters and SPI/I²C peripherals

Hardware Resources

* **MFRC522 RFID Reader Module Datasheet** – NXP PN532/RC522 reference, timing diagrams

* **HC-SR04 Ultrasonic Sensor Datasheet** – voltage levels, echo timing relationship

* **SG90 / TowerPro Micro Servo Datasheet** – pulse-width vs. angle chart, stall torque specs

* **16×2 Character LCD (HD44780) Datasheet** – command set, timing, busy-flag notes

* **PCF8574 I/O Expander Datasheet** – I2C address map, drive current limits

* **ESP328P Module Datasheet** – pinout, flash modes, Wi-Fi radio parameters

Export to PDF