**RFID-Based Parking System**

**🔖 Project Title:**

**RFID-Based Parking System with Gate Control and Access Status Display**

**🧾 Project Description:**

This project controls a gate barrier using an SG90 servo motor based on RFID authentication. A scanned RFID card is checked against a whitelist. If matched, the gate opens for a few seconds with access granted indication. If not, access is denied, and the gate remains closed. The system uses visual (LEDs), audible (buzzer), and textual (LCD) indicators to guide the user.

**🔩 Hardware Components:**

**(**[**URL:-https://app.cirkitdesigner.com/project/fb0600a0-c6e2-444f-b992-7a52e0ed14a9**](URL:-https://app.cirkitdesigner.com/project/fb0600a0-c6e2-444f-b992-7a52e0ed14a9)**)**

| **Component Name** | **Pin No** | **Destination Component** | **Pin No** | **Special Remark** |
| --- | --- | --- | --- | --- |
| Arduino Uno | 3 | Green LED (Anode) | + | Access Granted Indicator |
| Arduino Uno | 4 | Red LED (Anode) | + | Access Denied / Idle Indicator |
| Arduino Uno | 5 | Buzzer (+) | + | Beeping sound output |
| Arduino Uno | 6 | Servo Motor (SG90) | PWM | Controls gate barrier |
| Arduino Uno | 9 | RC522 RST | - | Reset pin of RFID module |
| Arduino Uno | 10 | RC522 SDA (SS) | - | Slave select pin |
| Arduino Uno | A4 (SDA) | I²C LCD SDA | - | I²C Communication |
| Arduino Uno | A5 (SCL) | I²C LCD SCL | - | I²C Communication |
| Arduino Uno | 11 | RC522 MOSI | - | SPI Communication |
| Arduino Uno | 12 | RC522 MISO | - | SPI Communication |
| Arduino Uno | 13 | RC522 SCK | - | SPI Clock |
| 5V / GND | - | All modules | - | Power Supply Common |

Ensure all GND pins from components are tied to Arduino GND.

**🧾 Libraries You Need**

* MFRC522
* SG90 Servo
* LiquidCrystal\_I2C

***Install them via the Arduino Library Manager (Sketch → Include Library → Manage Libraries).***

**🧾 AEDUINO CODE**

/\*

RFID‑BASED PARKING SYSTEM

─────────────────────────────────────────────────────────────────────────────

• Hardware : Arduino Uno + RC522 + 16×2 I²C LCD + SG90 Servo + LEDs + Buzzer

• Behaviour :

─ Idle : Red LED ON, buzzer beeps (150 ms ON / 350 ms OFF), gate closed,

LCD → “PLEASE SCAN UID TO PASS”.

─ Scan : LCD → “PLEASE WAIT – AUTHENTICATING…”.

─ Allowed : Two short beeps, green LED ON, gate opens 5 s, LCD “ACCESS GRANTED”.

─ Denied : One long beep, red LED remains ON, gate stays closed,

LCD “ACCESS DENIED”.

\*/

// ─────────────────────────── Libraries ──────────────────────────────────────

#include <SPI.h> // SPI bus for RC522

#include <MFRC522.h> // RFID reader driver

#include <Wire.h> // I²C bus for LCD

#include <LiquidCrystal\_I2C.h> // 16×2 I²C LCD

#include <Servo.h> // SG90 servo

// Replace .begin() with .init() so you can \*write\* init() everywhere.

#define init(...) begin(\_\_VA\_ARGS\_\_)

// ──────────────────────── Pin assignments ───────────────────────────────────

#define RST\_PIN 9 // RC522 reset

#define SS\_PIN 10 // RC522 SDA / SS

#define GREEN\_LED 3 // Green LED (access granted)

#define RED\_LED 4 // Red LED (idle / denied)

#define BUZZER 5 // Active buzzer

#define SERVO\_PIN 6 // SG90 servo signal

// ───────────────────────── Global objects ───────────────────────────────────

MFRC522 rfid(SS\_PIN, RST\_PIN); // RFID reader instance

LiquidCrystal\_I2C lcd(0x27, 16, 2); // LCD @ I²C address 0x27, 16 × 2

Servo gateServo; // Barrier arm servo

// ───────────── Whitelist of authorised UIDs (add more rows if needed) ───────

const char \*allowedUIDs[][5] = {

{"43","95","BB","F7"} // Example UID → "43 95 BB F7"

};

// ──────────────── Idle‑buzzer timing constants (non‑blocking) ───────────────

const unsigned long IDLE\_BEEP\_ON\_MS = 150; // Tone duration

const unsigned long IDLE\_BEEP\_OFF\_MS = 350; // Silence duration

// ────────────────── Runtime variables for idle buzzer ───────────────────────

unsigned long idleBeepTimestamp = 0; // Last toggle time

bool idleBuzzerState = false;// true = buzzer HIGH, false = LOW

// ──────────────── Forward declarations of helper functions ─────────────────

void idleScreen(); // Put system in idle state

void idleBeep(); // Generate pulsed buzzer in idle

bool isAllowed(const char \*); // Whitelist check

void grantAccess(); // Handle authorised card

void denyAccess(); // Handle unauthorised card

// ──────────────────────────── SETUP ───────────────────────────────────────

void setup() {

Serial.init(9600); // Serial Monitor for debugging

SPI.init(); // Start SPI bus

rfid.PCD\_Init(); // Initialise RC522

Wire.init(); // Start I²C bus

lcd.init(16, 2); // -> lcd.begin(16,2) (macro converts)

lcd.backlight(); // Turn on LCD back‑light

pinMode(GREEN\_LED, OUTPUT);

pinMode(RED\_LED, OUTPUT);

pinMode(BUZZER, OUTPUT);

gateServo.attach(SERVO\_PIN); // Attach servo to pin 6

gateServo.write(0); // Ensure gate is closed

idleScreen(); // Enter idle mode

Serial.println(F("System ready – waiting for card"));

}

// ──────────────────────────── LOOP ────────────────────────────────────────

void loop() {

// 1) If NO new card, stay in idle → keep buzzing every 500 ms

if (!rfid.PICC\_IsNewCardPresent() || !rfid.PICC\_ReadCardSerial()) {

idleBeep(); // Handle non‑blocking idle beep

return; // Restart loop

}

// 2) Card detected → stop idle beep and show "AUTHENTICATING"

digitalWrite(BUZZER, LOW); // Silence buzzer

idleBuzzerState = false; // Reset idle‑buzzer state machine

lcd.clear();

lcd.print("PLEASE WAIT");

lcd.setCursor(0,1);

lcd.print("AUTHENTICATING");

// 3) Build UID string like "43 95 BB F7"

char uidStr[15] = "";

for (byte i = 0; i < rfid.uid.size; i++) {

char byteHex[4];

sprintf(byteHex, "%02X", rfid.uid.uidByte[i]);

strcat(uidStr, byteHex);

if (i < rfid.uid.size - 1) strcat(uidStr, " ");

}

Serial.print(F("Scanned UID: "));

Serial.println(uidStr);

// 4) Compare with whitelist and act accordingly

if (isAllowed(uidStr)) grantAccess();

else denyAccess();

// 5) Release reader & return to idle

rfid.PICC\_HaltA();

rfid.PCD\_StopCrypto1();

idleScreen();

}

// ──────────────────────── Idle state helper ───────────────────────────────

void idleScreen() {

lcd.clear();

lcd.print("PLEASE SCAN UID");

lcd.setCursor(0,1);

lcd.print("TO PASS");

digitalWrite(RED\_LED, HIGH); // Red ON

digitalWrite(GREEN\_LED, LOW); // Green OFF

gateServo.write(0); // Gate closed

digitalWrite(BUZZER, LOW); // Start with buzzer LOW

idleBuzzerState = false;

idleBeepTimestamp = millis(); // Reset timer for pulsing

}

// ─────────── Non‑blocking idle buzzer (pulses every 500 ms) ───────────────

void idleBeep() {

unsigned long now = millis();

if (idleBuzzerState) { // Buzzer currently ON?

if (now - idleBeepTimestamp >= IDLE\_BEEP\_ON\_MS) {

digitalWrite(BUZZER, LOW); // Turn OFF

idleBuzzerState = false;

idleBeepTimestamp = now; // Reset timer

}

} else { // Buzzer currently OFF

if (now - idleBeepTimestamp >= IDLE\_BEEP\_OFF\_MS) {

digitalWrite(BUZZER, HIGH); // Turn ON

idleBuzzerState = true;

idleBeepTimestamp = now; // Reset timer

}

}

}

// ─────────────── Check if UID is in allowedUIDs list ──────────────────────

bool isAllowed(const char \*uid) {

const size\_t nCards = sizeof(allowedUIDs) / sizeof(allowedUIDs[0]);

for (size\_t i = 0; i < nCards; i++) {

char concat[15] = "";

for (byte b = 0; b < 4; b++) {

strcat(concat, allowedUIDs[i][b]);

if (b < 3) strcat(concat, " ");

}

if (!strcmp(uid, concat)) return true; // Match found

}

return false; // Not authorised

}

// ─────────────────── Handle authorised card ───────────────────────────────

void grantAccess() {

Serial.println(F("Access GRANTED"));

// Two short beeps (150 ms ON / 150 ms OFF)

for (byte i = 0; i < 2; i++) {

digitalWrite(BUZZER, HIGH); delay(150);

digitalWrite(BUZZER, LOW); delay(150);

}

digitalWrite(RED\_LED, LOW); // Turn off Red LED

digitalWrite(GREEN\_LED, HIGH); // Turn on Green LED

lcd.clear();

lcd.print("ACCESS GRANTED");

gateServo.write(90); // Open gate to 90°

delay(5000); // Keep open 5 seconds

gateServo.write(0); // Close gate

digitalWrite(GREEN\_LED, LOW); // Turn off Green LED

}

// ─────────────────── Handle unauthorised card ─────────────────────────────

void denyAccess() {

Serial.println(F("Access DENIED"));

digitalWrite(BUZZER, HIGH); delay(1000); // One long beep

digitalWrite(BUZZER, LOW);

digitalWrite(RED\_LED, HIGH); // Keep Red LED ON

digitalWrite(GREEN\_LED, LOW); // Ensure Green LED OFF

lcd.clear();

lcd.print("ACCESS DENIED");

delay(2000); // Display for 2 seconds

}

**📟 Display Messages:**

| **State** | **LCD Line 1** | **LCD Line 2** |
| --- | --- | --- |
| Idle | PLEASE SCAN UID | TO PASS |
| Authenticating | PLEASE WAIT | AUTHENTICATING |
| Access Granted | ACCESS GRANTED |  |
| Access Denied | ACCESS DENIED |  |

**🔊 Buzzer Patterns:**

| **State** | **Buzzer Behavior** |
| --- | --- |
| Idle | 150ms ON every 500ms |
| Access Granted | Two short beeps (150ms) |
| Access Denied | One long beep (1000ms) |

**🚦 LED Indicators:**

| **State** | **Green LED** | **Red LED** |
| --- | --- | --- |
| Idle | OFF | ON |
| Access Granted | ON (5s) | OFF |
| Access Denied | OFF | ON |

**🚪 Gate Servo Behavior:**

| **State** | **Servo Angle** | **Action** |
| --- | --- | --- |
| Idle / Denied | 0° | Gate Closed |
| Access Granted | 90° for 5 seconds | Gate Open |

You can change 90° to 180° depending on your mechanical setup.

**📶 Allowed RFID UIDs:**

You can add more authorized UIDs in this format:

const char \*allowedUIDs[][5] = {

{"43", "95", "BB", "F7"},

{"XX", "XX", "XX", "XX"}

};

Each UID is split byte by byte in hexadecimal (e.g., 43 95 BB F7).

**⚙️ Software Requirements:**

* Arduino IDE
* Libraries:
  + SPI.h
  + MFRC522.h
  + Wire.h
  + LiquidCrystal\_I2C.h
  + Servo.h

**🖥️ Serial Monitor Output:**

| **Event** | **Message** |
| --- | --- |
| System Boot | System ready – waiting... |
| UID Detected | Scanned UID: xx xx xx xx |
| Access Granted | Access GRANTED |
| Access Denied | Access DENIED |