RFID Based Vehicle Access Control System Code

- **CODE**

```
// RFID Vehicle Starter with 4-Wire LCD (16x2)
// Hardware: Arduino UNO, MFRC522 RC522, Relay, Buzzer, LED, 16x2 LCD
// Includes: prints UID to Serial + LCD, checks against authorized list

#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>

// RFID module pins
#define RST_PIN 9
#define SS_PIN  10

// Output pins
const uint8_t RELAY_PIN  = 7;
const uint8_t BUZZER_PIN = 6;
const uint8_t LED_PIN    = 5;

// Starter activation duration (ms)
const unsigned long START_DURATION_MS = 3000UL;

// LCD pin mapping: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);

// MFRC522 object
MFRC522 mfrc522(SS_PIN, RST_PIN);

// ----- Authorized UIDs (4-byte UIDs) -----
// Add authorized tags here. Each row is one UID (four bytes).
// To add more, add another row {0xAA,0xBB,0xCC,0xDD}
const byte AUTHORIZED_COUNT = 1;
const byte authorizedUIDs[AUTHORIZED_COUNT][4] = {
  { 0x31, 0xA7, 0xB4, 0x7B }   // <-- YOUR UID 43B77631
};

void setup() {
  Serial.begin(115200);
  delay(50);

  SPI.begin();
  mfrc522.PCD_Init();

  pinMode(RELAY_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
```

```
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(LED_PIN, LOW);

    // Initialize LCD
    lcd.begin(16, 2);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("RFID Starter");
    lcd.setCursor(0, 1);
    lcd.print("Initializing...");
    delay(1200);
    lcd.clear();
    lcd.print("Swipe the Card");

    Serial.println(F("RFID Vehicle Starter - Ready"));
  }

  void loop() {
    // Wait for a new card
    if (!mfrc522.PICC_IsNewCardPresent()) {
      return;
    }
    if (!mfrc522.PICC_ReadCardSerial()) {
      return;
    }

    // Read UID bytes and print to Serial & LCD
    byte uidLen = mfrc522.uid.size;        // usually 4 for common tags
    byte uid[10];                          // buffer for UID bytes
    String uidHexSpaced = "";
    String uidHexNoSpace = "";

    for (byte i = 0; i < uidLen; i++) {
      uid[i] = mfrc522.uid.uidByte[i];
      // print padded hex to Serial
      if (uid[i] < 0x10) Serial.print('0');
      Serial.print(uid[i], HEX);
      Serial.print(' ');

      // build strings
      if (uid[i] < 0x10) uidHexNoSpace += "0";
      uidHexNoSpace += String(uid[i], HEX);
      if (uid[i] < 0x10) uidHexSpaced += "0";
      uidHexSpaced += String(uid[i], HEX);
      uidHexSpaced += " ";
    }
    Serial.println();
    uidHexNoSpace.toUpperCase();
```

```
    uidHexSpaced.toUpperCase();

    // Show on LCD (line1: label, line2: UID no-space or spaced if small)
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Tag UID:");
    lcd.setCursor(0, 1);
    // if UID fits, show no-space (4 bytes = 8 chars). Otherwise show
  spaced.
    if (uidLen <= 4) {
      lcd.print(uidHexNoSpace);
    } else {
      lcd.print(uidHexSpaced);
    }

    // Check authorization
    bool ok = isAuthorized(uid, uidLen);

    delay(400); // short pause so user can see UID on LCD

    if (ok) {
      Serial.println(F("Authorized! Activating starter..."));
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Ignition Started");
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(RELAY_PIN, HIGH);   // trigger relay
      digitalWrite(BUZZER_PIN, LOW);
      delay(START_DURATION_MS);
      digitalWrite(RELAY_PIN, LOW);
      digitalWrite(LED_PIN, LOW);
      lcd.clear();
      lcd.print("Engine Started");
      delay(1100);
    } else {
      Serial.println(F("Unauthorized card! Access denied."));
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Access Denied!");
      digitalWrite(BUZZER_PIN, HIGH);
      delay(700);
      digitalWrite(BUZZER_PIN, LOW);
    }

    // Reset LCD prompt
    lcd.clear();
    lcd.print("Swipe the Card");

```

```
    // Stop reading and crypto
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();

    delay(300); // debounce
  }

  // Compare scanned uid[] to the authorized list
  bool isAuthorized(byte *uid, byte uidSize) {
    // This implementation expects 4-byte UIDs (common). If you have
  variable-length,
    // expand the table to include lengths or adapt logic.
    if (uidSize != 4) return false; // only handle 4-byte UIDs here

    for (byte i = 0; i < AUTHORIZED_COUNT; i++) {
      bool match = true;
      for (byte j = 0; j < 4; j++) {
        if (uid[j] != authorizedUIDs[i][j]) {
          match = false;
          break;
        }
      }
      if (match) return true;
    }
    return false;
  }
```