RELATÓRIO FINAL: SISTEMA DE GERENCIAMENTO DE ESTACIONAMENTO COM NFC E SENSOR INFRAVERMELHO

Luis henrique ferracciu pagotto mendes,* Maria Eduarda Soares Romana Silva,† Murilo fontana muniz‡

Emails: luismendes.2020@alunos.utfpr.edu.br, silva.2003@alunos.utfpr.edu.br, murilo.2022@alunos.utfpr.edu.br

Abstract— This report details the design and implementation of a prototype for an automated parking management system. The main objective is to control vehicle entry and exit, providing real-time monitoring of available parking spaces. The system employs a Near Field Communication (NFC) reader to grant access, where an authorized tag triggers a servo motor that simulates a barrier gate. Upon entry, the system decrements the count of available spots, which starts at a total of ten. Vehicle exit is detected by an infrared presence sensor, which in turn opens the gate and increments the vacancy count. The number of available spots is continuously displayed on an LCD screen connected via an I2C module, offering a clear and immediate user interface. This project demonstrates the feasibility of a microcontroller-based solution for smart parking management, proving to be a functional, low-cost, and efficient prototype.
Keywords— Microcontroller Systems; Parking Management; NFC; Servo Motor; Presence Sensor; I2C.

Resumo— O presente relatório demonstra o planejamento e a criação de um modelo inicial para um sistema automatizado de administração de estacionamento. A meta central é supervisionar o fluxo de veículos que entram e saem, acompanhando em tempo real o número de espaços livres. O sistema emprega um leitor de *Near Field Communication* (NFC) para regular o acesso, ativando um servomotor que representa uma barreira ao detectar uma etiqueta válida. Cada vez que um carro entra, o sistema diminui em um o total de dez lugares. A saída dos carros é percebida por um sensor infravermelho, que ordena a abertura da barreira e aumenta a contagem de vagas desocupadas. O número de vagas disponíveis é mostrado de forma contínua em um visor LCD com módulo I2C, fornecendo uma interface direta e compreensível para o usuário. O projeto exemplifica a praticidade de uma solução fundamentada em microcontrolador para a gestão inteligente de estacionamentos, confirmando ser um protótipo viável, acessível e produtivo.

Palavras-chave— Sistemas Microcontrolados; Gerenciamento de Estacionamento; NFC; Servo Motor; Sensor de Presença; I2C.

1 Introdução

Este documento apresenta um protótipo funcional para a gestão automatizada de estacionamentos, desde sua concepção até a efetiva aplicação. A meta central é simplificar o processo de entrada e saída de carros, além de acompanhar a situação das vagas em tempo real.

Para identificar os veículos ao entrarem, o sistema emprega a tecnologia NFC (*Near Field Communication*). Basta aproximar a etiqueta NFC do leitor para que o acesso seja confirmado, uma vaga seja marcada como ocupada e um motor simule a abertura da barreira. Na saída, um sensor identifica a presença do veículo, liberando a passagem e atualizando o número de vagas disponíveis.

Um painel de LED, controlado por um módulo I2C, mostra quantas vagas estão livres, criando uma forma simples e direta de informação para todos. O objetivo é mostrar que um sistema microcontrolado pode ser uma solução inteligente e eficaz para gerenciar estacionamentos, aproveitando melhor o espaço e tornando a vida dos usuários mais fácil.

2 Materiais e Métodos

Para a construção do protótipo, foram utilizados os seguintes componentes de hardware e ferramentas de software:

2.1 Componentes de Hardware

- Microcontrolador Arduino Uno;
- Módulo Leitor NFC/RFID RC522;
- Tag NFC (13.56MHz);
- Servo Motor SG90;
- Sensor de Presença Infravermelho (IR-KY032);
- *Display* LCD 16x2 com Módulo I2C;
- *Protoboard* e *Jumpers* para as conexões;
- Cabo USB A/B;
- Potenciômetro para controle de luminosidade do *Display* LCD;

2.2 Software e Ferramentas

- Arduino IDE para desenvolvimento do código;
- *Overleaf* para a documentação do projeto;
- Bibliotecas (*Libraries*) específicas para os módulos utilizados:

- `MFRC522.h` para o leitor NFC;
- `Servo.h` para o servo motor;
- `LiquidCrystal.h` para o *display* LCD I2C;

2.3 Metodologia de Implementação

A montagem do sistema seguiu as etapas abaixo:

1. **Montagem do Circuito:** Conexão dos componentes na *protoboard* de acordo com o esquemático elétrico definido. O microcontrolador foi posicionado como a unidade central, conectando-se a cada um dos periféricos (sensores, atuador e *display*).

2. **Desenvolvimento do código:** O código foi programado na Arduino IDE. A lógica implementa a leitura da tag NFC para a entrada, o acionamento da cancela (servo motor), a detecção de saída pelo sensor de presença e a atualização constante do *display* com o número de vagas.

3. **Testes e Validação:** Foram realizados testes unitários e integrados. Testou-se a leitura da tag, a precisão do sensor de presença, o movimento do servo e a correta exibição das informações no *display*. A contagem de vagas, iniciando em 10, foi validada em múltiplos ciclos de entrada e saída.

3 Código Fonte

O código do sistema foi desenvolvido em C++ utilizando a IDE do Arduino. O código-fonte principal está apresentado abaixo. Ele é responsável por inicializar os componentes, gerenciar a lógica de controle de vagas e interagir com os periféricos.

Listing 1: Código principal do firmware do sistema de estacionamento.

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

void atualizarDisplayPrincipal();
void verificarEntradaPorTag();
void verificarSaida();
void acionarCancelaComPulso(int ledPin);

// --- PINOS ---
#define RFID_SS_PIN      10
#define RFID_RST_PIN     9
#define SERVO_PIN        A0
#define SENSOR_SAIDA_PIN 2

#define LCD_RS 6
#define LCD_E  5
#define LCD_D4 4
#define LCD_D5 3
#define LCD_D6 8
#define LCD_D7 7

#define LED_ENTRADA A1
#define LED_SAIDA   A2

// --- CONFIGURAcoES ---
const int TOTAL_VAGAS        = 10;
const unsigned long DEBOUNCE = 2500;
const unsigned long PULSO_MS = 2000;
const unsigned long MOV_MS   = 1500;

String UID_AUTORIZADA = "BC 09 0A 4A";
int vagasDisponiveis   = TOTAL_VAGAS;
unsigned long lastRFID = 0;
String lastUID         = "";

MFRC522 rfid(RFID_SS_PIN, RFID_RST_PIN);
Servo cancela;
LiquidCrystal lcd(LCD_RS, LCD_E, LCD_D4,
    LCD_D5, LCD_D6, LCD_D7);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();

  // LCD
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Estacionamento");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(500);

  // Servo fechado
  cancela.attach(SERVO_PIN);
  cancela.write(0);

  // LEDs start OFF (pino HIGH -> LED apaga no
      seu esquema)
  pinMode(LED_ENTRADA, OUTPUT);
  pinMode(LED_SAIDA,   OUTPUT);
  digitalWrite(LED_ENTRADA, HIGH);
  digitalWrite(LED_SAIDA,   HIGH);

  // Sensor
  pinMode(SENSOR_SAIDA_PIN, INPUT);

  atualizarDisplayPrincipal();
  Serial.println("Sistema Pronto!");
}

void loop() {
  verificarSaida();
  verificarEntradaPorTag();
}

// ENTRADA por RFID
void verificarEntradaPorTag() {
  if (!rfid.PICC_IsNewCardPresent() ||
      !rfid.PICC_ReadCardSerial()) return;

  // monta string UID
  String uid = "";
  for (byte i=0; i<rfid.uid.size; i++) {
    uid += (rfid.uid.uidByte[i]<0x10 ? " 0" : "
        ");
    uid += String(rfid.uid.uidByte[i], HEX);
  }
  uid.toUpperCase();
  uid.remove(0, 1);

  unsigned long now = millis();
  if (uid==lastUID && now-lastRFID<DEBOUNCE) {
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
    return;
  }
  lastRFID = now;
  lastUID  = uid;

  Serial.print("Tag detectada: [");
      Serial.print(uid); Serial.println("]");

  if (uid==UID_AUTORIZADA && vagasDisponiveis >
      0) {
    Serial.println("Acesso Liberado");
    lcd.clear();
    lcd.print("Acesso Liberado!");
    vagasDisponiveis--;
    acionarCancelaComPulso(LED_ENTRADA);
    atualizarDisplayPrincipal();
  }
  else if (uid==UID_AUTORIZADA) {
    Serial.println("Lotado");
    lcd.clear();
    lcd.print("Lotado!");
```

```
    atualizarDisplayPrincipal();
  }
  else {
    Serial.println("Negado");
    lcd.clear();
    lcd.print("Acesso Negado!");
    delay(2000);
    atualizarDisplayPrincipal();
  }

  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}

// ÍSADA por sensor
void verificarSaida() {
  if (digitalRead(SENSOR_SAIDA_PIN)!=LOW)
      return;

  //lcd.clear();
  if (vagasDisponiveis<TOTAL_VAGAS) {
    vagasDisponiveis++;
    Serial.println("Carro saiu.");
    lcd.clear();
    lcd.print("Saida Liberada!");
    lcd.setCursor(0, 1);
    lcd.print("Volte Sempre!");

    acionarCancelaComPulso(LED_SAIDA);
    atualizarDisplayPrincipal();
  }
  delay(800); // debounce
}

// Mostra vagas no LCD
void atualizarDisplayPrincipal() {
  lcd.clear();
  lcd.print("Vagas Livres:");
  lcd.setCursor(0, 1);
  if (vagasDisponiveis<10) lcd.print('0');
  lcd.print(vagasDisponiveis);
}

// Substitui movimento + LED em uma mesma
    rotina bloqueante,
// mas garante que servo e LED partam juntos
void acionarCancelaComPulso(int ledPin) {
  // Liga LED (no seu wiring, LOW -> acende)
  digitalWrite(ledPin, LOW);

  // Movimento suave ate 90graus e volta em
      MOV_MS total
  int passos = 90;
  int dir    = 1;  // sobe primeiro
  int delayStep = (MOV_MS/2) / passos;
  for (int ang=0; ang<=90; ang++) {
    cancela.write(ang);
    delay(delayStep);
  }
  // segura no topo o restante do tempo
  delay(PULSO_MS - (MOV_MS/2));

  // volta (outros MOV_MS/2)
  for (int ang=90; ang>=0; ang--) {
    cancela.write(ang);
    delay(delayStep);
  }

  // Desliga LED
  digitalWrite(ledPin, HIGH);
}
```

3.1 Descrição Detalhada do Código

O software, apresentado na Listagem 1, foi criado usando C++ para ser executado no Arduino. A forma como ele foi pensado visa ser flexível e ágil, assegurando que cada parte do hardware conectado funcione bem. A seguir, encontra-se uma explicação completa de como ele está organizado e suas funções.

3.1.1 Estrutura Geral e Configurações Iniciais

O código é iniciado com a inclusão das bibliotecas necessárias para a comunicação com cada componente de hardware: `<SPI.h>` e `<MFRC522.h>` para o leitor RFID; `<Servo.h>` para o servo motor; e `<LiquidCrystal.h>` para o *display* LCD em modo de conexão paralela.

No cabeçalho, são definidas constantes e variáveis globais importantes:

- **Mapeamento de Pinos:** Todos os pinos de conexão dos componentes são definidos usando a diretiva `#define`. Isso centraliza a configuração do *hardware*, facilitando futuras modificações.
- **Parâmetros de Operação:** Constantes como `TOTAL_VAGAS`, o tempo de `DEBOUNCE` para o leitor RFID e a `UID_AUTORIZADA` para o acesso são pré-definidas.
- **Instância dos Objetos:** São criados os objetos `rfid`, `cancela` e `lcd`, que serão usados para controlar os respectivos componentes.

Na função `setup()`, o sistema é inicializado: a comunicação serial é ativada para depuração, os periféricos são iniciados, a cancela é posicionada no estado fechado (ângulo 0) e o *display* LCD exibe uma mensagem de boas-vindas antes de mostrar a tela principal de contagem de vagas.

3.1.2 *Loop* Principal e Lógica de Eventos

O núcleo do programa reside na função `loop()`, que atua de maneira direta e eficaz. Ela invoca, sem parar, as duas funções de checagem essenciais: `verificarEntradaPorTag()` e `verificarSaida()`. Tal método assegura que o sistema se mantenha sempre apto a reagir a qualquer evento de entrada ou saída, evitando interrupções desnecessárias.

3.1.3 Detalhamento das Funções Principais

`verificarEntradaPorTag()`: Esta função gerencia todo o processo de entrada de veículos.

1. **Detecção e Leitura:** Verifica se uma nova tag foi aproximada do leitor e, em caso afirmativo, lê seu Identificador Único (UID), formatando-o em uma *String* para comparação.
2. **Mecanismo Anti-Repetição (*Debounce*):** Para evitar que uma única aproximação da tag acione a cancela várias vezes, o sistema ignora leituras repetidas da mesma tag dentro de um intervalo de 2,5 segundos.
3. **Lógica de Validação:** O UID lido é comparado com a `UID_AUTORIZADA`. Se a tag for válida e

houver vagas, o acesso é liberado, o contador de vagas é decrementado e a cancela é acionada. Caso contrário, mensagens de "Lotado!" ou "Acesso Negado!" são exibidas.

`verificarSaida()`: Responsável por controlar a saída de veículos, esta função monitora o sensor infravermelho. Ao detectar um objeto (sinal em nível `LOW`), ela incrementa o contador de vagas (se não estiver no máximo), exibe uma mensagem de despedida e aciona a cancela.

`acionarCancelaComPulso(int ledPin)`: Esta função auxiliar é responsável por criar um ciclo completo de operação da cancela de forma suave e sinalizada. Ela recebe como parâmetro o pino do LED a ser acionado (entrada ou saída). O servo motor move-se gradualmente de 0 a 90 graus, permanece aberto por um tempo determinado e depois retorna suavemente à posição inicial, enquanto o LED correspondente fica aceso durante todo o processo.

`atualizarDisplayPrincipal()`: Uma rotina simples que limpa o *display* LCD e exibe a contagem atualizada de "Vagas Livres:", garantindo que o usuário sempre tenha a informação correta. Ela também formata a contagem para sempre exibir dois dígitos (ex: 09, 08, etc.).

4 Circuito Implementado

O circuito elétrico do protótipo foi montado em uma *protoboard* para facilitar as conexões e os testes, como mostra a Figura 1. A Figura 2 apresenta o diagrama esquemático das ligações entre o microcontrolador e os demais componentes.

montado.png

Figura 1: Imagem do Projeto Montado na *Protoboard*

As principais conexões são:

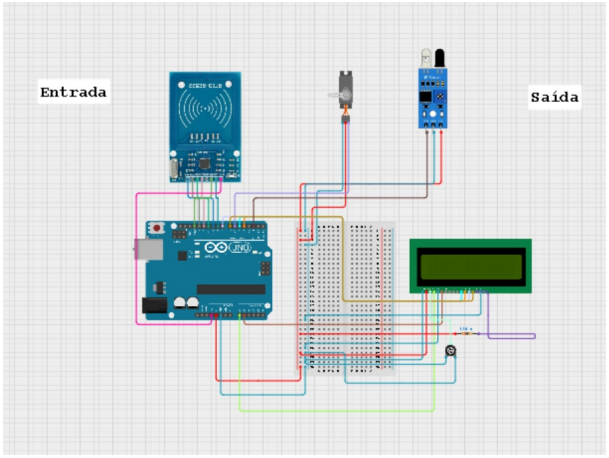

Figura 2: Diagrama esquemático do sistema de estacionamento.

- **Leitor NFC (MFRC522):** Conectado via interface SPI ao microcontrolador.
- **Servo Motor:** O pino de sinal foi conectado a uma porta PWM (*Pulse-Width Modulation*) do microcontrolador.
- **Sensor de Presença:** Conectado a uma porta digital configurada como entrada.
- ***Display* LCD I2C:** Conectado aos pinos SDA (dados) e SCL (*clock*) da interface I2C do microcontrolador.

5 Conclusão

O desenvolvimento deste trabalho permitiu consolidar os conhecimentos adquiridos na disciplina de Sistemas Microcontrolados, aplicando-os em um problema prático e de relevância atual. O projeto do sistema de gerenciamento de estacionamento demonstrou ser funcional e eficaz em cumprir os requisitos propostos.

A utilização do leitor NFC para entrada e do sensor infravermelho para saída mostrou-se uma solução robusta e de baixo custo para o controle de fluxo. A exibição em tempo real das vagas disponíveis no *display* LCD oferece uma interface clara e útil para o usuário final.

Para trabalhos futuros, sugere-se a integração do sistema a uma plataforma em nuvem (IoT) para monitoramento remoto, registro de dados históricos e a possibilidade de reserva de vagas via aplicativo mobile. Adicionalmente, o sistema pode ser expandido para controlar múltiplos acessos e saídas, tornando-o escalável para estacionamentos de maior porte.