B.S. Park – Sistema de Estacionamento Inteligente

Bernardo Silva (2020112296)
Simão Dias (2020132169)

Projeto efetuado no âmbito da Unidade Curricular Internet das Coisas

Mestrado em Análise de Dados e Sistemas de Apoio à Decisão

Professor: André Rodrigues

Coimbra, junho de 2025

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUÇÃO

No âmbito da unidade curricular de **Internet das Coisas** do Mestrado de **Análise de Dados e Sistemas de Apoio à Decisão**, lecionada pelo Professor André Rodrigues, foi proposto, ao longo das aulas, o desenvolvimento de um projeto prático. Esta atividade teve como principal objetivo aplicar e consolidar os conhecimentos adquiridos durante a unidade curricular, através da concretização de um sistema funcional com base num tema previamente sugerido pelo docente.

Após uma análise criteriosa das hipóteses disponíveis e possíveis ideias a desenvolver, a nossa escolha recaiu sobre o desenvolvimento de um **sistema de controlo de acesso a um parque de estacionamento**, uma solução que representa um desafio interessante no contexto da **Internet das Coisas**, ao envolver a integração entre sensores, atuadores e lógica de programação para controlo de entradas e saídas de veículos.

De forma a viabilizar o projeto, o docente forneceu diversos equipamentos, incluindo microcontroladores e componentes eletrónicos, que nos permitiram construir uma maquete funcional.

Além disso, foi disponibilizada uma base de código inicial que serviu de ponto de partida. Este código foi progressivamente adaptado às necessidades específicas do nosso sistema, ao utilizarmos a linguagem ***MicroPython***.

Este relatório descreve as várias etapas do projeto, desde o planeamento até à implementação e testes finais, onde se destacam as decisões técnicas, os desafios enfrentados e os resultados obtidos.

2. DEFINIÇÃO DO TEMA DE PROJETO

Para a realização deste projeto, o docente disponibilizou três propostas distintas, das quais selecionámos a opção número 2. Esta opção enquadrou-se no desenvolvimento de um **sistema de controlo de acesso a um parque de estacionamento**, com abertura automática da cancela mediante autenticação por **RFID/NFC**, **deteção de proximidade**, e **visualização em tempo real** do estado do parque.

O sistema desenvolvido foi concebido para permitir o acesso de veículos ao parque de forma automatizada, com recolha e monitorização de dados que são apresentados num ***dashboard*** em tempo real. Além do controlo físico (abertura/fecho da cancela), o sistema também indica, num pequeno ecrã **OLED**, o número de lugares disponíveis (entre outras informações), simulado na maquete.

No final, foi utilizada a plataforma ***ThingsBoard*** para analisar os dados recolhidos e facilitar a visualização e o acompanhamento do estado do sistema à **distância**.

O presente projeto permitiu aplicar conceitos fundamentais da **Internet das Coisas**, como a comunicação entre sensores e atuadores, a utilização de microcontroladores, e a integração com plataformas de visualização e análise de dados.

3. MATERIAIS E EQUIPAMENTOS UTILIZADOS

Para implementar o sistema descrito, foram disponibilizados e utilizados os seguintes componentes:

Material	**Referência**
Expansion Board	*Expansion Board Base for* XIAO *with Grove* OLED
ESP32	*Seed Studio* XIAO ESP32-S3 *Sense*
Breadboard	*Adafruit Breadboard*
Leitor RFID/NFC	PN532 NFC
Sensor de luz e proximidade	*Adafruit* APDS9960
Servo (cancela)	*TowerPro* SG-5010
Cabo para *breadboard*	*Adafruit Qwiic to male*

Todos os componentes foram ligados ao microcontrolador e a programas necessários (**IDE *Thonny***) que permitiram utilizar a linguagem ***MicroPython***, permitindo assim uma gestão eficiente dos recursos e uma rápida prototipagem do sistema.

4. DESENHO DA SOLUÇÃO

Após a definição do tema e seleção dos componentes necessários, iniciou-se o processo de montagem do sistema na *breadboard*, com o objetivo de testar e validar o funcionamento de cada componente individualmente, antes de integrá-los num sistema completo.

O microcontrolador utilizado, ***Seed Studio* XIAO ESP32-S3 *Sense***, possui conectividade *Wi-Fi* e capacidade de processamento suficiente para gerir os sensores, atuadores e comunicar com o ***dashboard***. A ***breadboard*** foi usada como base para realizar todas as ligações elétricas de forma prática, uma vez que permite modificações rápidas durante os testes.

Entre os componentes integrados, destacam-se:

- O **leitor RFID/NFC,** utilizado para autenticar os utilizadores com cartões;
- O **sensor de luz e proximidade,** para deteção da presença do veículo na zona de estrada;
- O **servo motor**, responsável por simular a abertura e fecho da cancela;
- O **ecrã OLED**, incorporado na *expansion board*, que mostra informações como o número de lugares disponíveis;
- Os **cabos e resistências** que foram essenciais para assegurar as ligações entre todos os módulos.

Durante esta fase, todos os testes foram realizados com código desenvolvido em ***Micropython.***

A montagem inicial do circuito pode ser observada na figura seguinte:

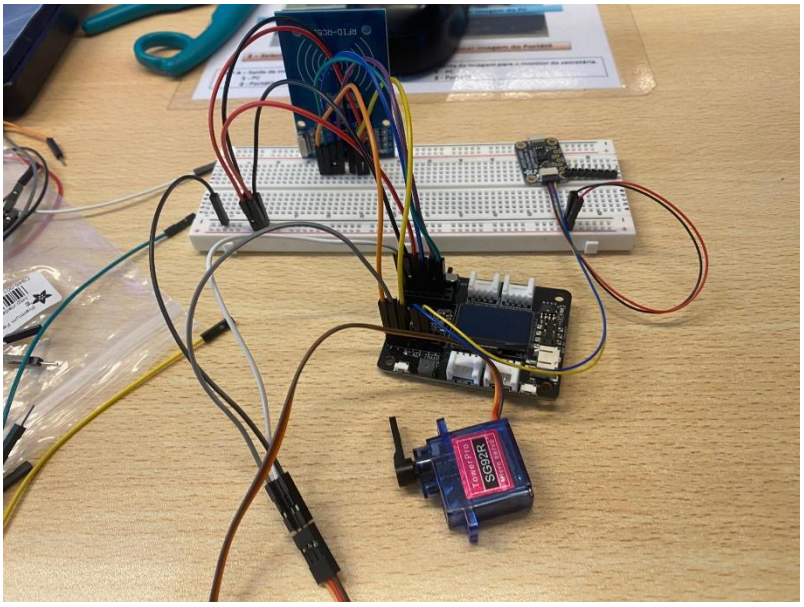

Figura 1: Montagem Inicial do Circuito na *Breadboard*

5. DESENVOLVIMENTO DO PROJETO

Após a montagem física da maquete e a validação individual dos componentes, iniciou-se a implementação funcional do sistema, tanto a nível de ***hardware*** como de ***software***, ao utilizar-se o ambiente ***Thonny*** com a linguagem ***Micropython***.

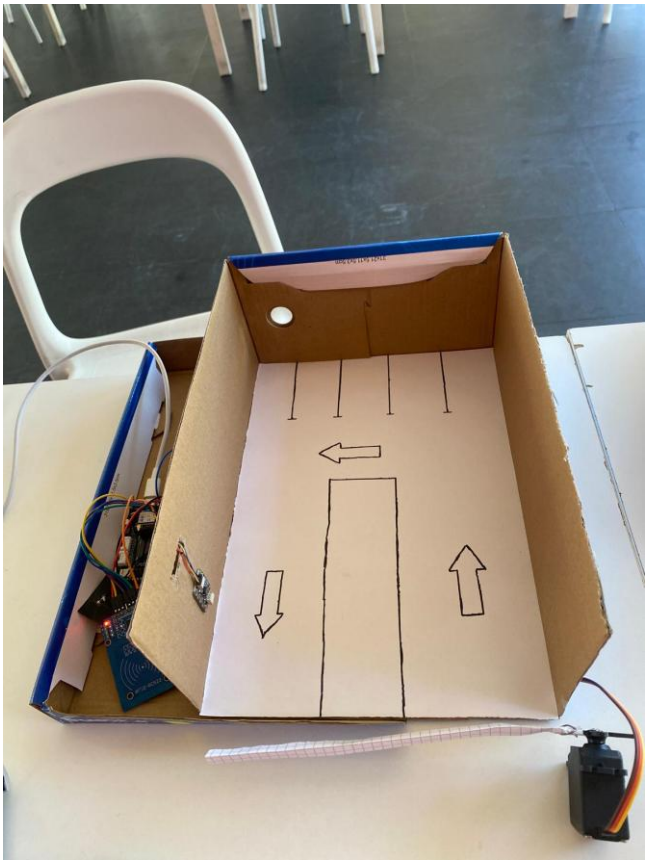

Figura 2: 1ª Fase de Montagem da Maquete

O projeto foi concebido para simular o controlo de acesso a um parque de estacionamento com uma única cancela partilhada entre a entrada e a saída de veículos. A entrada de veículos é realizada através da leitura de cartões **RFID** através do módulo **PN532**, onde apenas um dos cartões foi previamente registado como válido. Este controlo permite simular cenários de **acesso autorizado, acesso negado** ou **falta de lugares disponíveis**, consoante o a **taxa de ocupação** do parque. Em cada um destes casos, são ativados ***feedbacks* sonoros distintos** através de um ***buzzer*** e são apresentadas mensagens correspondentes no **ecrã *OLED*** integrado.

Em **estado de espera** *(Standby)*, o *display* mostra continuamente o **número de lugares disponíveis** no parque. Quando um **cartão válido** é aproximado do leitor e há lugares livres, o sistema autoriza a entrada, abre a cancela controlada por um **servo motor**, regista a hora de entrada sincronizada **via *NTP*** (com ajuste para a hora de Lisboa), e gera automaticamente um ***ID* único para cada entrada**, com base no ***UID* do cartão** e num contador interno. Estes registos são armazenados em memória numa estrutura tipo histórico.

```
# ========== INICIALIZAÇÃO ==========
i2c = SoftI2C(scl=Pin(6), sda=Pin(5))
oled = SSD1306_I2C(128, 64, i2c)
apds9960 = APDS9960LITE(i2c)
apds9960.prox.enableSensor()
rdr = MFRC522(sck=7, mosi=9, miso=8, rst=3, cs=1)
servo = Servo(pin_id=2)
buzzer = PWM(Pin(4, Pin.OUT))

entradas = 0
saidas = 0
portao_aberto = False
tempo_ultimo_acesso = 0
tempo_aberto_ms = 5000
historico = []
entradas_ativas = {}
uid_contadores = {}

# ========== FUNÇÕES ==========
def beep(duration=200, freq=1000):
    buzzer.freq(freq)
    buzzer.duty(512)
    time.sleep_ms(duration)
    buzzer.duty(0)

def beep_acesso_negado():
    for _ in range(3):
        beep(100, 3000)
        time.sleep_ms(100)

def beep_sem_lugar():
    beep(500, 2500)
    time.sleep_ms(200)
    beep(150, 2000)
    time.sleep_ms(100)
    beep(150, 2000)

def beep_thank_you():
    beep(100, 1500)
    time.sleep_ms(100)
    beep(150, 1000)

def mostrar_oled(linhas):
    oled.fill(0)
    for i, linha in enumerate(linhas):
        oled.text(linha, 0, i * 10)
    oled.show()
    try:
        client.publish("v1/devices/me/attributes", json.dumps({"mensagem_oled": " | ".join(linhas)}))
    except Exception as e:
        print("Erro ao enviar mensagem OLED para ThingsBoard:", e)

def abrir_portao():
    global portao_aberto, tempo_ultimo_acesso
    servo.write(90)
    portao_aberto = True
    tempo_ultimo_acesso = time.ticks_ms()

def fechar_portao():
    global portao_aberto
    servo.write(0)
    portao_aberto = False

def lugares_ocupados():
    return entradas - saidas

def lugares_livres():
    return TOTAL_LUGARES - lugares_ocupados()

def enviar_json(dados):
    try:
        client.publish("v1/devices/me/telemetry", json.dumps(dados))
    except Exception as e:
        print("Erro ao enviar MQTT:", e)
```

Figura 3: Código (Inicialização e Funções)

Para a simulação da **saída dos veículos**, o sistema utiliza um **sensor de proximidade** (APDS9960*)*. Quando é detetada a presença de um veículo e existem vários veículos no parque, é selecionado **aleatoriamente** um dos **IDs** em aberto para **processar a saída**. Com base na hora atual e na hora de entrada previamente guardada, o sistema calcula automaticamente o **tempo de permanência** em minutos e o **custo correspondente**, usando uma taxa fixa de 0,50 €/minuto. Estes dados são exibidos no ecrã *OLED* e registados para posterior análise.

O sistema foi também desenhado para ser **autónomo e robusto**, com verificação de estados e controlo automático de cancela (a abrir e a fechar após um tempo pré-definido – 5 segundos). Foram ainda incluídas **funções de *beep*** personalizadas para diferenciar situações como: **acesso negado**, **parque cheio**, **entrada autorizada** ou **agradecimento na saída**.

O código faz uso das bibliotecas ***machine, time, network, ntptime*** e ***umqtt.simple*** para lidar com os **pinos GPIO**, **temporizações**, **ligação *Wi-Fi*, sincronização horária e envio de dados via MQTT**, para posterior visualização no *ThingsBoard*, abordada no próximo ponto.

```
# ===== SCRIPT ATUALIZADO COM ESTATÍSTICAS E MENSAGENS OLED NO THINGSBOARD =====
import time
import random
import network
import ntptime
import json
import usocket as socket

from machine import Pin, SoftI2C, PWM, RTC
from mfrc522 import MFRC522
from ssd1306 import SSD1306_I2C
from servo import Servo
from apds9960LITE import APDS9960LITE
from umqtt.simple import MQTTClient
```

Figura 4: Bibliotecas Utilizadas

6. ANÁLISE E VISUALIZAÇÃO DOS RESULTADOS

Para permitir uma análise clara e em tempo real do funcionamento do nosso sistema de controlo de acesso ao parque de estacionamento, optou-se por utilizar a plataforma ***ThingsBoard***, à qual foram enviados os dados recolhidos através do protocolo ***MQTT***, previamente configurado no código em ***Micropython***.

```
# ========== CONFIGURAÇÕES ==========
TOTAL_LUGARES = 5
PRECO_POR_MINUTO = 0.5
THINGSBOARD_HOST = "eu.thingsboard.cloud"
ACCESS_TOKEN = "EKBknhKTtyw2CyTZxv9A"
WIFI_SSID = "simaodias"
WIFI_PASSWORD = "12345678"
cartoes_autorizados = [
    [183, 214, 5, 1, 101]
]

# ========== CONECTAR AO WIFI ==========
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_SSID, WIFI_PASSWORD)
while not wlan.isconnected():
    time.sleep(1)
print("Conectado ao Wi-Fi")

# ========== HORA VIA NTP ==========
ntptime.host = "pool.ntp.org"
try:
    ntptime.settime()
except:
    print("Falha ao sincronizar hora NTP")
rtc = RTC()

def hora_atual_str():
    tm = time.localtime(time.time() + 3600)  # UTC+1
    return "{:02}:{:02}:{:02}".format(tm[3], tm[4], tm[5])

def segundos_entre(dt1, dt2):
    t1 = dt1[3]*3600 + dt1[4]*60 + dt1[5]
    t2 = dt2[3]*3600 + dt2[4]*60 + dt2[5]
    return max(t2 - t1, 1)

# ========== MQTT ==========
client = MQTTClient("client_id", THINGSBOARD_HOST, user=ACCESS_TOKEN, password="")
client.connect()
print("Conectado ao ThingsBoard via MQTT")
```

Figura 5: Conexão ao *Wi-Fi,* Hora Atual, MQQT e Envio de Dados para o *ThingsBoard*.

A nossa *interface* no *ThingsBoard* foi estruturada de forma a permitir uma leitura rápida dos principais indicadores do sistema. Os dados enviados incluem:

- **UID** dos **cartões** utilizados;
- **Identificadores únicos das entradas**;
- **Hora de entrada e de saída** de cada veículo;
- **Custo total** de permanência;
- **Tempo de permanência** em minutos;
- **Número de lugares disponíveis**;
- **Número de entradas e saídas**;
- **Total de faturação**;
- **Tentativas de entrada inválidas** e por **parque cheio**;
- **Hora atual sincronizada**;
- **Objetivo mensal** (definido hipoteticamente para controlo).

Com base nestes dados, construiu-se o ***dashboard*** com os seguintes ***widgets***:

a) Logótipo e Identidade
No canto superior esquerdo foi inserido o logótipo oficial "B.S. PARK", que reforça a identidade visual do projeto.

b) Indicadores Gerais
- Lugares disponíveis – *Widget* do tipo *Gauge* mostra em tempo real os lugares livres;
- Número total de entradas – Contabiliza o número total de entradas validadas;
- Tentativas com parque cheio – Contabiliza quantas vezes foi feito um pedido de entrada sem disponibilidade;
- Tentativas com cartões inválido**s** – Mostra tentativas de entrada não autorizadas.

c) Visão Comercial e Operacional
- Total de faturação – Calculado automaticamente com base no tempo de permanência e tarifa definida (0,50€/min);
- Objetivo mensal – Valor de referência para controlo de desempenho;
- Comparação de entradas válidas vs acessos negados – Gráfico de barras que permite visualizar os picos de movimento e falhas de acesso.

d) Tabelas e Registos
- Registo de entradas e saídas – *Widget* do tipo tabela com colunas como *Timestamp*, Tipo e ID;
- *Card* OLED – Mostra em tempo real a mensagem atual do ecrã *OLED* da maquete.

e) Hora e Horário
- Hora atual – Indicador com a hora sincronizada;
- Horário do parque – Informação visual das horas de funcionamento.

f) Tempo médio de permanência
- Gráfico de linha – Mostra o tempo de permanência médio dos veículos ao longo do dia.

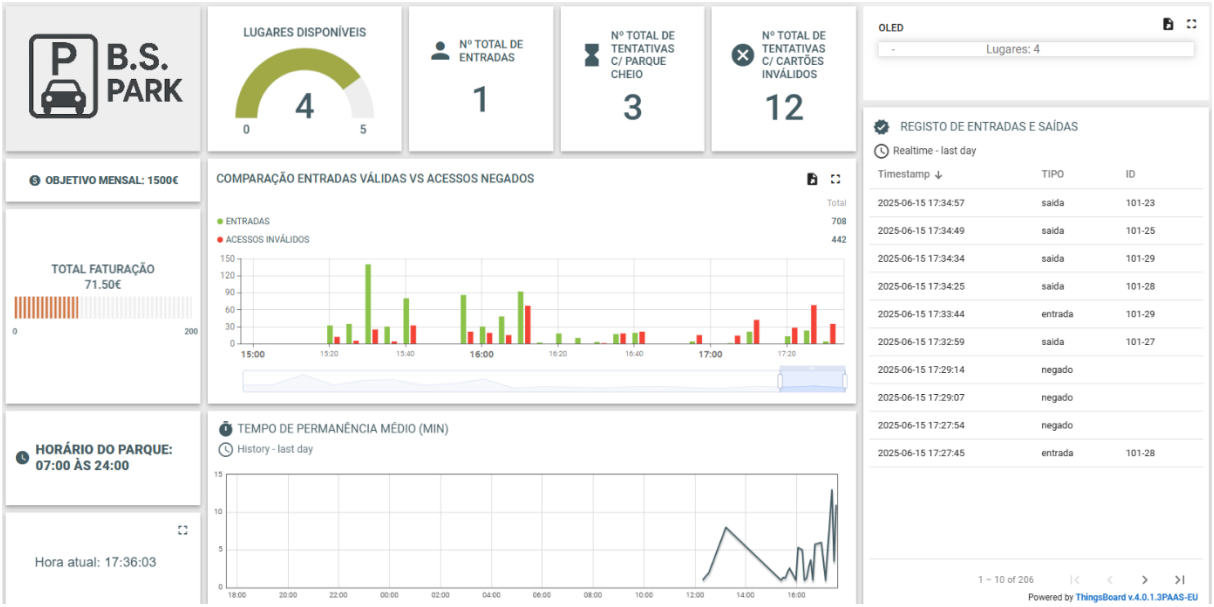

Figura 6: *Dashboard* Final

Estas ferramentas não só permitem validar o funcionamento e a lógica do sistema, como também apresentam uma camada de análise essencial para monitorizar e avaliar o desempenho do parque em tempo real.

A utilização de **identificadores únicos**, o **cálculo automático do custo** por **tempo de permanência** e a **organização visual clara** tornam a solução eficaz, escalável e alinhada com os princípios da **Internet das Coisas.**

7. CONCLUSÃO

Este projeto permitiu desenvolver um sistema funcional de controlo de acesso a um parque de estacionamento, aplicando assim conceitos fundamentais da **Internet das Coisas** (IoT). Através da integração entre *hardware* (sensores, atuadores, leitor RFID, ecrã *OLED*) e *software* (programação em *MicroPython* e comunicação via *MQTT*), foi possível criar uma maquete que simula com realismo a entrada e saída de veículos, com registo automático dos dados e visualização em tempo real.

Durante o desenvolvimento enfrentou-se desafios relacionados com a leitura dos cartões *RFID*, sincronização horária e lógica de controlo da cancela, mas conseguiu-se superar com pesquisa, testes e trabalho em equipa. A geração de IDs únicos para cada entrada, o cálculo do custo de permanência com base no tempo, e os diferentes *feedbacks* sonoros para cada cenário foram detalhes que tornaram o sistema mais robusto e realista.

A integração com o ***ThingsBoard*** representou um ponto-chave do projeto, ao permitir a monitorização remota do sistema e a visualização dos dados num ***dashboard*** interativo. Esta etapa não só reforçou os conhecimentos de comunicação em **IoT** como também forneceu uma perspetiva prática sobre a importância da **análise de dados em tempo real** para a gestão de recursos físicos.

Por fim, este projeto constituiu uma **experiência rica e desafiante**, proporcionando um contacto direto com tecnologias emergentes e aproximando-nos da realidade dos **sistemas inteligentes aplicados à mobilidade urbana**. Conseguimos, com sucesso, aplicar conhecimentos teóricos de forma prática, culminando na construção de uma solução completa, eficaz e com potencial de escalabilidade para contextos reais.