# Praia Inteligente

Eduardo Jorge Marques Antunes - 2240850

Rodrigo dos Santos Silva - 2240817

Trabalho de Projeto da unidade curricular de Tecnologias de Internet

Leiria, Junho de 2025

# Lista de Figuras

# Lista de siglas e acrónimos

| | |
|---|---|
| **DHT** | Digital Humidity and Temperature |
| **ESTG** | Escola Superior de Tecnologia e Gestão |
| **IPLeiria** | Instituto Politécnico de Leiria |
| **LDR** | Light Dependent Resistor |
| **LED** | Light Emitting Diode |
| **MCU** | Microcontroller Unit (arduino) |
| **SBC** | Single-Board Computer (raspberry) |

# Índice

# 1. Introdução

## 1 – Objeto do trabalho (Praia Inteligente)

No âmbito da unidade curricular de Tecnologias de Internet do curso de Engenharia Informática, desenvolveu-se o projeto "Praia Inteligente", concebido em duas fases com o propósito de aplicar competências Web-IoT.O projeto integra um Arduino MKR1000 e um Raspberry Pi 3 ligados a uma interface Web responsiva, capaz de recolher e monitorizar dados em tempo-real do estado ambiental da praia, sinalizar informação crucial para a segurança balnear e registar todo o histórico de eventos.

## 2 – Justificação / pertinência

A disponibilização de informação para o melhor conforto e segurança em zonas balneares depende de supervisão constante, informação fidedigna e avisos rápidos sobre condições meteorológicas, sinalização devida e possíveis perigos. Falhas de sinalização ou atrasos na atualização dos diferentes estados da praia podem colocar em risco banhistas e dificultar o trabalho de nadadores-salvadores. A Praia Inteligente responde a estes desafios através de:

Atualização automática do estado da iluminação: o LED de iluminação (Arduíno) acende quando a LDR do Raspberry indica pouca luz, cortando custos energéticos ao desligar-se em plena luz do dia;

Alerta térmico: um LED no Raspberry acende quando a temperatura da água ultrapassa 20 °C, sugerindo boas condições para os mais tímidos;

Sinalização sonora e visual: Um LED vermelho liga-se e um buzzer soa continuamente enquanto a bandeira estiver vermelha, chamando a atenção imediata dos banhistas e pessoal de apoio assim;

Disponibilização de botões que permitem facilitar o trabalho dos nadadores-salvadores na transição das bandeiras, ao ativar bandeira vermelha disponibiliza também de uma camara que tira fotos, criando evidência fotográfica do evento.

## 3 – Objetivos do trabalho

Projetar e implementar uma plataforma integrada (hardware + software) que monitorize, registe e atue sobre variáveis ambientais de uma praia em tempo-real, proporcionando uma sinalização intuitiva e de fácil acesso através e uma interface Web.

**Principais capacidades da Praia Inteligente:**

- Monitorização da temperatura da água, velocidade do vento e se está de dia ou de noite;
- Gestão remota da sinalização das bandeiras (verde/amarela/vermelha), disponibilizando, também em caso de bandeira vermelha sinalização sonora e visual.
- Captura automática de fotografias quando a bandeira vermelha está hasteada.
- Consulta através de um painel Web, de toda a informação da praia incluindo se esta está a ser vigiada, a percentagem de ocupação da mesma, assim como todos outros dados referidos anteriormente
- Manter um histórico de todas as leituras e ações;

## 4 – Métodos e técnicas utilizados

- Hardware - Arduíno MKR1000 e Raspberry Pi 3

**Arduíno MKR1000:**

- Sensor: sensor DHT11 (medição de temperatura/humidade) , no projeto substituimos humidade por velocidade do vento e como o valor da humidade é muito alto temos que vento = (humidade / 3)w
- Atuadores: LED para sinalizar Dia/Noite, LED sinalizador de bandeira vermelha.

**Raspberry Pi 3:**

- Sensor: LDR para classificar se está de Dia/Noite
- Atuadores: Buzzer sinalizador auditivo para bandeira vermelha, LED de sinalização para temperatura da água acima de 20ºC, semáforo para as Bandeiras (LEDs vermelho / amarelo / verde) e câmara USB.

**Sensores e atuadores:**

DHT11: lê temperatura/humidade → envia por Wi-Fi.

LDR: classifica Dia/Noite → API POST.

LEDs / Buzzer / semáforo: comandados via GPIO consoante condições.

**Firmware:**

Código C++ (Arduino) para leitura do DHT, controlo de LEDs e pedidos GET/ PUT ao servidor.

Python (Raspberry) para leitura do LDR, acionamento do buzzer/semáforo e publicações POST.

**Dashboard:**

Disponibilização de todos os dados da praia e de 3 botões que permitem o nadador-salvador alterar a bandeira, secção de histórico filtrável e página de câmara que exibe e arquiva a fotografia captada no momento da bandeira vermelha.

# 2. Arquitetura

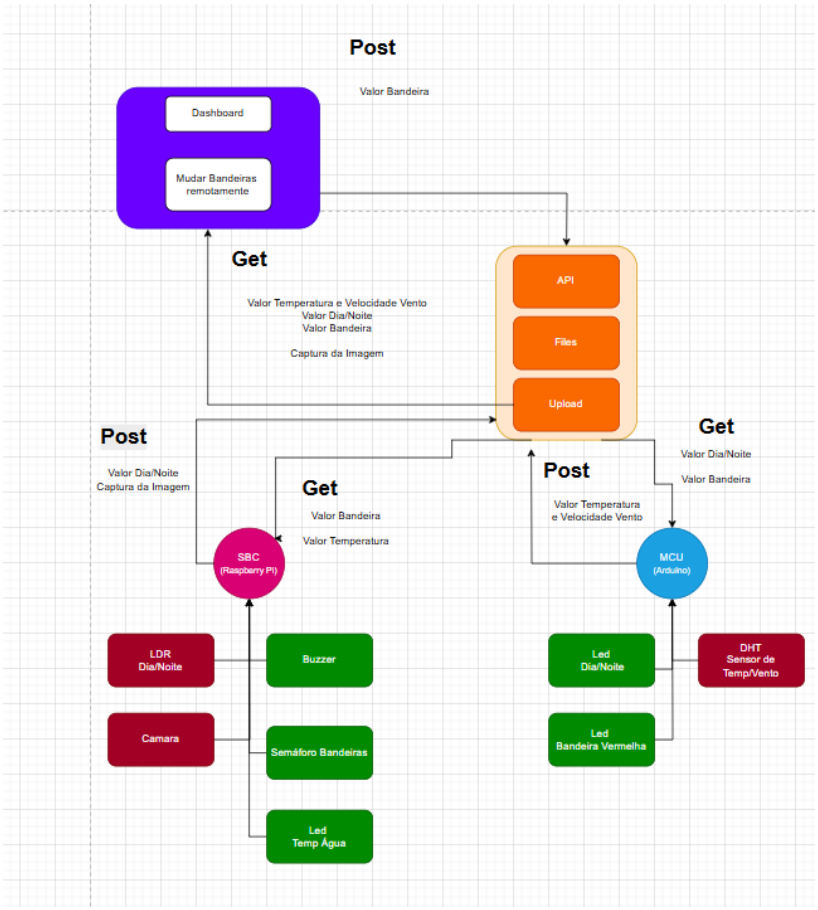



**Figura 1- Praia Inteligente IOT**

### Componentes

Sensor de temperatura e humidade (DHT) :

O valor utlizado no projeto é o do vento que é quivale á humidade medida a dividir por 3 . Ambos o vento e a temperatura vão sendo atualizados na dashboard através de um post.

Sensor LDR :

Verifica a luz do local , só devolve 0 ou 1 , o 0 equivale a ter muita luz (Dia) e o 1 pouca luz (Noite)

LEDs :

Usamos 3 leds , dois no Rasperry e um no Arduíno cada um a funcionar em função dos gets feitos na api .

Buzzer:

O buzzer utlizado liga quando o valor da bandeira é vermelha

Semáforo :

Um semáforo com as cores verde amarela e vermelha que acendem dependendo do valor da bandeira .

Unidades de Controlo MCU (Arduíno)

• Recebe dados dos sensores de temperatura e humidade .

• Controla os LEDs de Dia e Noite e o LED da bandeira vermelha.

• Envia os dados para a API utilizando o método POST.

• Obtém comandos e dados da API através de GET.

SBC (Raspberry Pi)

• Controla o sensor LDR .

• Envia os valores do sensor LDR para a API via POST.

• Vai buscar o valor das bandeiras via GET.

• Controla o buzzer consoante o valor da bandeira e também acende ou desliga o LED consoante o valor da temperatura .

API

- É o centro de comunicação entre todos os dispositivos e o sistema.
- Disponibiliza endpoints para:
- GET: Obtenção de dados (temperatura, vento, luminosidade )
- POST: Envio de dados e comandos (por exemplo, ligar o buzzer quando a bandeira está vermelha )

Dashboard (Interface de Utilizador)

- Interface gráfica onde o utilizador pode:
- Mudar o valor da bandeira através de botoes
- Visualizar a camara numa página separada.
- Comunica-se com a API usando os métodos GET e POST.

Fluxo de Comunicação

1. Os sensores enviam dados para o MCU (Arduino) ou o SBC (Raspberry Pi).

2. Estes dispositivos comunicam com a API usando pedidos HTTP.

3. A Dashboard envia comandos e consulta dados através da API.

4. A API responde e envia as instruções adequadas aos dispositivos físicos.

# 3. Implementação

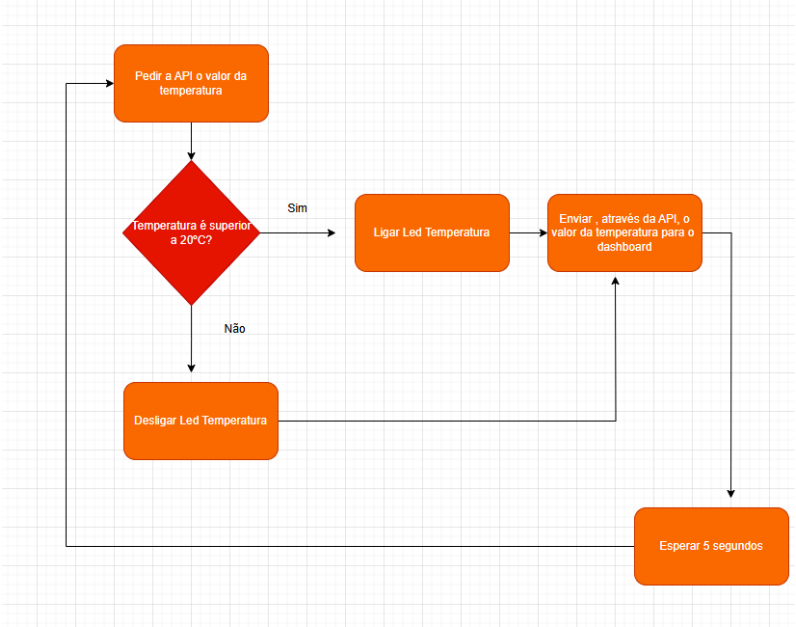



**Figura 5 - Exemplo de um diagrama para representação da Temperatura**

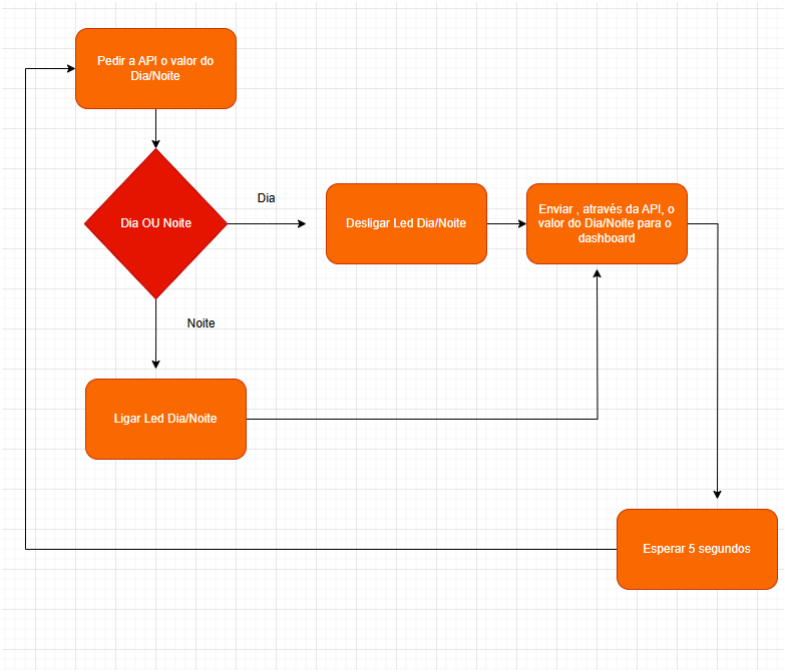



**Figura 6 - Exemplo de um diagrama para representação do Dia/Noite**

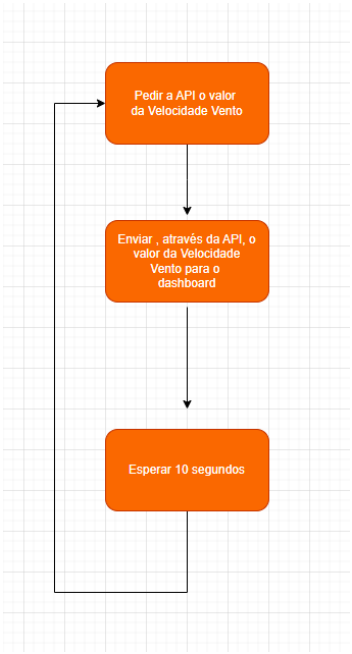



**Figura 7 - Exemplo de um diagrama para representação da Velocidade do Vento**

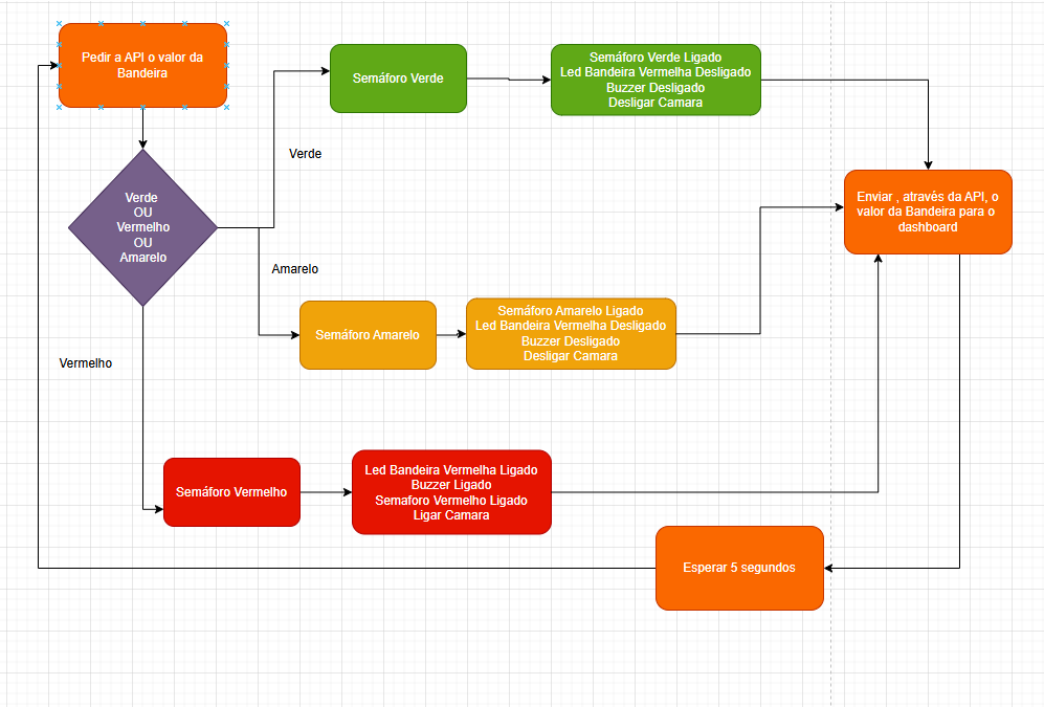



**Figura 8 - Exemplo de um diagrama para representação do Valor Bandeira**

# 4. Cenário de Teste

### ARDUINO

Incializaçao do codigo :

```
char PASSWORD[] = "1nv3nt@r2023_IPLEIRIA";
char HOST[]     = "iot.dei.estg.ipleiria.pt";
int  PORTO      = 80;

WiFiClient wifi;
HttpClient clienteHTTP(wifi, HOST, PORTO);

// ============================
// Definição de pinos
// ============================
#define DHTPIN          0    // Pino conectado ao DHT11
#define DHTTYPE       DHT11
#define LED_DIA_NOITE   6    // LED indicador Dia/Noite
#define LED_BANDEIRA    7    // **LED indicador de bandeira vermelha**

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(LED_DIA_NOITE, OUTPUT);
  pinMode(LED_BANDEIRA, OUTPUT);   // **Configura pino da bandeira**

  // Conecta ao WiFi
  Serial.print("Ligando ao WiFi");
  WiFi.begin(SSID, PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConectado ao WiFi!");
}
```

Loop com delay no final

gd

```
void loop() {
  // --- 1) Leitura do DHT11 ---
  float temperatura = dht.readTemperature();
  float humidade    = dht.readHumidity();
  float ventoFinal  = humidade / 3.0;

  if (isnan(temperatura) || isnan(humidade)) {
    Serial.println("Erro ao ler do DHT11");
    delay(2000);
    return;
  }

  Serial.print("Temperatura: ");
  Serial.print(temperatura, 1);
  Serial.print(" °C, Humidade: ");
  Serial.print(humidade, 1);
  Serial.println(" %");

  // --- 2) Envio de Dados via POST ---
  String dados = "";
  dados += "valor_Tempagua=" + String(temperatura, 1);
  dados += "&data_Tempagua=2025-06-14%2012:00:00";
  dados += "&nome_Tempagua=Tempagua";
  dados += "&valor_Vento="    + String(ventoFinal, 1);
  dados += "&data_Vento=2025-06-14%2012:00:00";
  dados += "&nome_Vento=Vento";

  clienteHTTP.post(
    "https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api.php",
    "application/x-www-form-urlencoded",
    dados
```

```
  // --- 3) GET Dia/Noite ---
  clienteHTTP.get(
    "https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api.php?nome=Dia_Noite"
  );
  String respDN = clienteHTTP.responseBody();
  Serial.print("GET Dia/Noite → ");
  Serial.println(respDN);

  // --- 4) Controle do LED Dia/Noite ---
  if (respDN.indexOf(": Noite") >= 0) {
    digitalWrite(LED_DIA_NOITE, HIGH);
    Serial.println("LED Dia/Noite → ACESO");
  } else {
    digitalWrite(LED_DIA_NOITE, LOW);
    Serial.println("LED Dia/Noite → APAGADO");
  }

  // --- 5) GET Bandeira ---
  clienteHTTP.get(
    "https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api.php?nome=Bandeira"
  );
  String respBandeira = clienteHTTP.responseBody();
  Serial.print("GET Bandeira → ");
  Serial.println(respBandeira);
```

```
  // --- 6) Controle do LED da Bandeira ---
  // Supondo resposta algo como "Valor da bandeira: Vermelha"
  if (respBandeira.indexOf("Vermelha") >= 0) {
    digitalWrite(LED_BANDEIRA, HIGH);
    Serial.println("LED Bandeira → ACESO (Vermelha)");
  } else {
    digitalWrite(LED_BANDEIRA, LOW);
    Serial.println("LED Bandeira → APAGADO");
  }

  // --- 7) Aguarda antes de reiniciar o loop ---
  delay(10000);
}
```

**Rasperry :**

Setup dos pinos e imports necessarios :

```
import requests

import RPi.GPIO as GPIO

from time import sleep, strftime, localtime

from datetime import datetime

import cv2




webcam_url="http://10.20.229.104:4747/video"

# ============================

# Configuração de pinos GPIO

# ============================

# Sensor LDR (Dia/Noite)

LDR_PIN       = 18

# LED para temperatura

TEMP_LED_PIN  = 23

# LEDs das bandeiras

PIN_VERMELHO  = 17

PIN_VERDE     = 27
```

```
# Buzzer de 3 pinos (VCC, GND e SINAL) → usar um GPIO livre, ex. BCM 24
BUZZER_PIN    = 24

# Modo BCM e desliga avisos
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

# Seção LDR
GPIO.setup(LDR_PIN, GPIO.IN)
# Seção Temperatura
GPIO.setup(TEMP_LED_PIN, GPIO.OUT)
# Seção Bandeiras
GPIO.setup(PIN_VERMELHO, GPIO.OUT)
GPIO.setup(PIN_VERDE, GPIO.OUT)
GPIO.setup(PIN_AMARELO, GPIO.OUT)
# Seção Buzzer
GPIO.setup(BUZZER_PIN, GPIO.OUT)
```

```
# Estado inicial: tudo desligado
GPIO.output(TEMP_LED_PIN, GPIO.LOW)
GPIO.output(PIN_VERMELHO, GPIO.LOW)
GPIO.output(PIN_VERDE, GPIO.LOW)
GPIO.output(PIN_AMARELO, GPIO.LOW)
GPIO.output(BUZZER_PIN, GPIO.LOW)

# Teste rápido do buzzer para garantir hardware OK
GPIO.output(BUZZER_PIN, GPIO.HIGH)
sleep(2)
GPIO.output(BUZZER_PIN, GPIO.LOW)
```

Agora algumas funções auxiliares para organizar um pouco mais o main. .

```
# ==============================
# Funções Auxiliares
# ==============================
def apagar_leds_bandeira():
    GPIO.output(PIN_VERMELHO, GPIO.LOW)
    GPIO.output(PIN_VERDE, GPIO.LOW)
    GPIO.output(PIN_AMARELO, GPIO.LOW)
```

```
def acender_cor(cor):
    apagar_leds_bandeira()
    if cor == "vermelha":
        GPIO.output(PIN_VERMELHO, GPIO.HIGH)
    elif cor == "verde":
        GPIO.output(PIN_VERDE, GPIO.HIGH)
    elif cor == "amarela":
        GPIO.output(PIN_AMARELO, GPIO.HIGH)
    else:
        apagar_leds_bandeira()
```

```python
def pegar_cor_da_api(url):
    try:
        resposta = requests.get(url, params={"nome": "Bandeira"})
        if resposta.status_code == 200:
            texto = resposta.text.lower()
            print("Resposta da API (bandeira):", texto)
            if "verde" in texto:
                return "verde"
            elif "vermelha" in texto:
                return "vermelha"
            elif "amarela" in texto:
                return "amarela"
            else:
                print("Cor não reconhecida na resposta.")
        else:
            print("Erro na requisição bandeira:", resposta.status_code)
    except Exception as e:
        print("Erro ao conectar na API (bandeira):", e)
    return None
```

LoopPrincipal:

```python
def main():
    url_api = "https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api.php"

    while True:
        agora = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        # --- Seção 1: LDR (Dia/Noite) POST ---
        valor_ldr = GPIO.input(LDR_PIN)
        estado_ldr = "Dia" if valor_ldr == GPIO.LOW else "Noite"
        print(f"[{agora}] LDR → {estado_ldr}")
        dados = {
            "valor_Dia": estado_ldr,
            "data_Dia": agora,
            "nome_Dia": "Sensor LDR"
        }
        try:
            resp = requests.post(url_api, data=dados)
            print(f"[{agora}] POST Dia/Noite → {resp.status_code}")
```

```
189  def main():
195      while True:
225          except Exception as e:
226
227              print("Erro no POST Dia/Noite:", e)
228
229
230
231          # --- Seção 2: Temperatura GET + LED ---
232
233          try:
234
235              r = requests.get(url_api, params={"nome": "Tempagua"})
236
237              r.raise_for_status()
238
239              temp = float(r.text.strip().split()[-1])
240
241              print(f"[{agora}] Tempagua → {temp}°C")
242
243              if temp > 20:
244
245                  GPIO.output(TEMP_LED_PIN, GPIO.HIGH)
246
247                  print("LED Temperatura → ACESO")
248
249              else:
250
251                  GPIO.output(TEMP_LED_PIN, GPIO.LOW)
252
253                  print("LED Temperatura → APAGADO")
254
255          except Exception as e:
256
257              print("Erro no GET Tempagua ou controlo do LED:", e)
```

```
def main():
    while True:


        # --- Seção 3: Bandeiras GET + LED + Buzzer ---

        cor = pegar_cor_da_api(url_api)

        if cor:

            print(f"[{agora}] Bandeira → {cor}")

            acender_cor(cor)

            # Buzzer ON apenas se Vermelha

            if cor == "vermelha":

                GPIO.output(BUZZER_PIN, GPIO.HIGH)

            else:

                GPIO.output(BUZZER_PIN, GPIO.LOW)

        else:

            print(f"[{agora}] Nenhuma cor válida recebida, apagando LEDs de bandeira.")

            apagar_leds_bandeira()

            GPIO.output(BUZZER_PIN, GPIO.LOW)
```

```python
    # --- Seçao 4: Captura de imagem se Vermelha ---
    # Aguarda 10 segundos para próxima leitura

    sleep(10)

    if cor == "vermelha":

            cap = cv2.VideoCapture(webcam_url)

            ret, frame = cap.read()

            if ret:

                cv2.imwrite('captura.jpg', frame)

            cap.release()


            url = 'https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api/upload.php'

            files = {'imagem': open('captura.jpg', 'rb')}

            m = requests.post(url, files=files)


            hora = strftime("%Y-%m-%d %H:%M:%S", localtime())

            requests.post('https://iot.dei.estg.ipleiria.pt/ti/ti169/PROJETO%20PRAIA%20INTELIGENTE/PROJETO/api/files/Camera/data.txt', hora)


    else:
```

```python
def main():
    while True:

        else:

            print("Erro na requisição:", r.status_code)


# ============================

# Execução e limpeza

# ============================

try:

    main()

except KeyboardInterrupt:

    print("Encerrando...")

finally:

    # Desliga tudo antes de sair

    GPIO.output(TEMP_LED_PIN, GPIO.LOW)

    apagar_leds_bandeira()

    GPIO.output(BUZZER_PIN, GPIO.LOW)

    GPIO.cleanup()

    print("Programa terminado.")
```

## Implementaçao codigo arduino :

1. **Leitura do sensor DHT11**

   O programa inicia executando a leitura da temperatura em graus Celsius e da humidade relativa em percentagem a partir do sensor DHT11. Em seguida, calcula-se um valor "ventoFinal" dividindo a humidade por 3, apenas como exemplo de processamento adicional. Se qualquer uma das leituras resultar em valor inválido (NaN), o sistema escreve "Erro ao ler do DHT11" no monitor série, faz uma pausa de dois segundos e interrompe aquela iteração.

2. **Impressão no monitor série**

   Após garantir que as leituras são válidas, o sistema formata os dados para uma casa decimal e transmite ao monitor série a mensagem "Temperatura: X.X °C, Humidade: Y.Y %", permitindo verificar em tempo real os valores obtidos pelo DHT11.

3. **Envio de dados via HTTP POST**

   Em seguida, constrói-se uma cadeia de caracteres no formato application/x-www-form-urlencoded contendo os parâmetros nomeados "valor_Tempagua", "data_Tempagua" (com data e hora fixas no exemplo), "nome_Tempagua", "valor_Vento", "data_Vento" e "nome_Vento". Esses dados são então enviados ao servidor remoto através de uma requisição POST. O código de status da resposta HTTP (por exemplo, 200) e o corpo da resposta são exibidos no monitor série, confirmando o sucesso ou indicando falha.

4. **Obtenção do estado Dia/Noite via HTTP GET**

   O dispositivo faz uma requisição GET ao mesmo endpoint, passando o parâmetro "nome=Dia_Noite". A resposta, que pode ser algo como "Valor: Noite", é armazenada em memória e também mostrada no monitor série com a indicação "GET Dia/Noite → …".

5. **Controle do LED de Dia/Noite**
   Com base no texto retornado, verifica-se se há a palavra “Noite”. Caso positivo, o LED associado ao indicador Dia/Noite é acionado; caso contrário, permanece desligado. Cada mudança de estado também é registrada no monitor série.

6. **Obtenção e controle da bandeira**
   De forma análoga, o programa faz outra requisição GET, agora com “nome=Bandeira”. Se a resposta contiver “Vermelha”, o LED correspondente à bandeira fica aceso, indicando bandeira vermelha; em outro caso, esse LED é desligado. Mais uma vez, o resultado aparece no monitor série.

Fluxo geral

O código implementa um ciclo típico de aplicação IoT: ler sensores locais, enviar dados para um servidor, consultar parâmetros externos e acionar atuadores (LEDs) conforme instruções recebidas. Dessa forma, combina medições in loco com lógica remota para oferecer um sistema de monitorização e sinalização integrado.

**RASPERRY**

- Configuração dos pinos GPIO
  É definido o modo de numeração BCM e desativados os avisos do sistema. Seguem-se as instruções para cada pino: o sensor LDR como entrada, os LEDs de temperatura e de bandeira como saídas, e o buzzer também como saída. Imediatamente após, garante-se que todos os atuadores (LEDs e buzzer) estão desligados, escrevendo nível lógico baixo em cada um.
- Teste do buzzer
  Antes de iniciar o ciclo contínuo, ativa-se o buzzer durante dois segundos para verificar o hardware, desligando-o em seguida. Esta breve rotina serve para

assegurar que o sinal sonoro está funcional e que futuros acionamentos não falharão silenciosamente.

- Funções auxiliares

  – apagar_leds_bandeira(): desliga simultaneamente os três LEDs de cor, limpando qualquer estado anterior.

  – acender_cor(cor): invoca primeiro a limpeza dos LEDs e depois acende apenas aquele cujo identificador corresponde à string recebida (“vermelha”, “amarela” ou “verde”), garantindo exclusividade de indicação.

  – pegar_cor_da_api(url): realiza uma requisição GET ao servidor pedindo o parâmetro “Bandeira”. Se a resposta vier com sucesso (código 200), converte o texto para minúsculas e procura pelas palavras-chave “verde”, “vermelha” ou “amarela”. Em caso de correspondência, devolve o nome da cor; em caso de falha na rede ou resposta não reconhecida, imprime mensagem de erro e retorna `None`.

- Loop principal

  Dentro de um laço infinito, com timestamp atualizado a cada iteração, o programa faz:

  • LDR (Dia/Noite) POST: lê o valor digital do sensor (baixo = dia, alto = noite), prepara um dicionário com o estado, data/hora e nome do sensor, e envia via POST ao endpoint. O código de resposta é registado ou, em falha, imprime erro de conexão.

  • Temperatura GET + LED: obtém por GET o valor de “Tempagua”, extrai o número da resposta, imprime-o e acende o LED de temperatura se exceder 20 °C; caso contrário, mantém-no apagado. Eventuais exceções de rede ou parsing são capturadas e reportadas.

  • Bandeira GET + LED + Buzzer: chama pegar_cor_da_api para receber a cor;

acende o LED correspondente e liga o buzzer apenas se a cor for “vermelha”. Se não houver cor válida, garante que LEDs e buzzer fiquem desligados. • Captura de imagem se bandeira vermelha: após aguardar 10 s, caso a última cor tenha sido vermelha abre-se a stream da webcam, captura-se um frame e grava-se em arquivo. Em seguida, envia-se a imagem para o servidor via POST multipart e regista-se o timestamp num ficheiro remoto também por POST.

- Encerramento e limpeza
  O laço principal é executado dentro de um bloco que trata `KeyboardInterrupt` para permitir parar o programa com Ctrl+C. No bloco `finally`, antes de sair, desliga-se todos os LEDs e o buzzer e chama-se `GPIO.cleanup()`, restaurando o estado dos pinos e evitando conflitos em execuções futuras.

Equipamento utlizado :

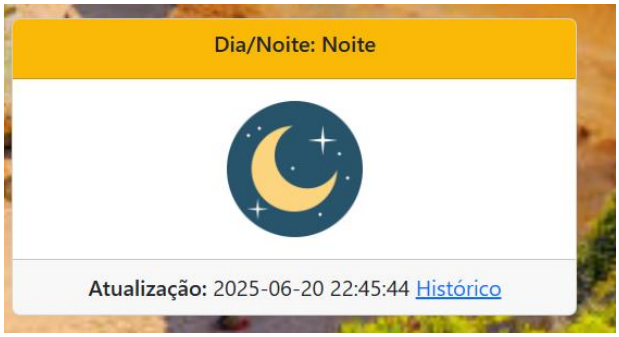
- Raspberry Pi (com conector de rede Wi-Fi ou Ethernet)
- Sensor DHT11 (temperatura e humidade)
- Sensor LDR (fotocélula)
- LEDs:
  - 1 LED para indicação de temperatura
  - 1 LED para indicação se esta de dia ou noite
- Semáforo com uma cor para cada bandeira
- Buzzer de 3 pinos (VCC, GND e sinal)
- Webcam (streaming via IP)
- Cabos de ligação e resistores adequados
- Fonte de alimentação para o Raspberry Pi

# 5. Resultados obtidos

Agora mostrando os resultados .

Quando o LDR lê pouca luz :

- Nesta situação o led do Arduíno irá acender



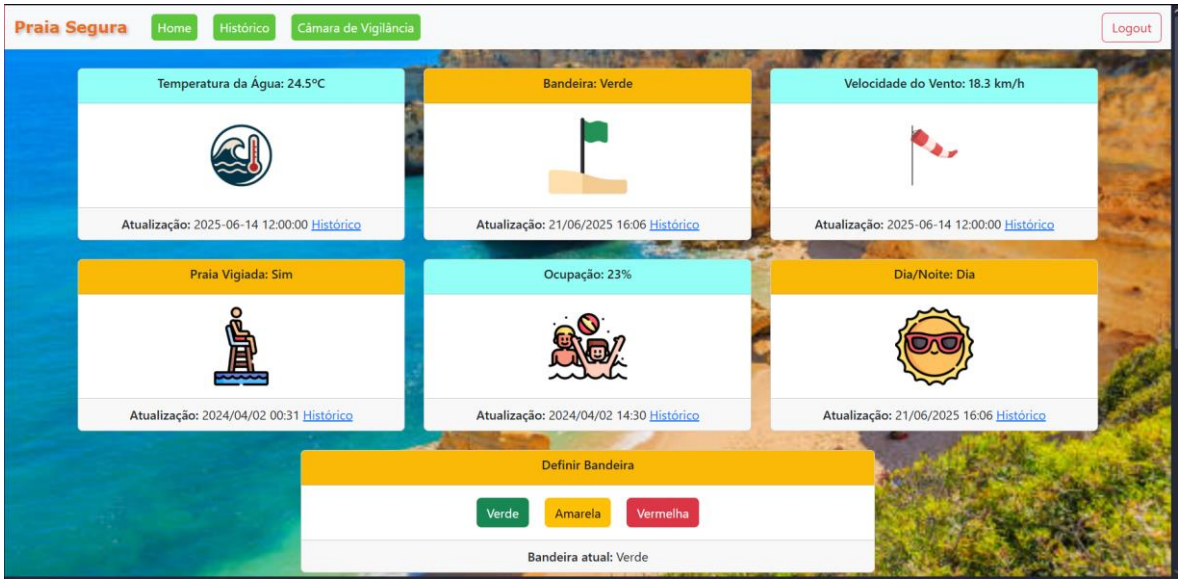

Quando o LDR lê muita luz :

- Nesta situação p led do Arduíno irá estar apagado


Agora apresentamos os botões que alteram os valores da bandeira :

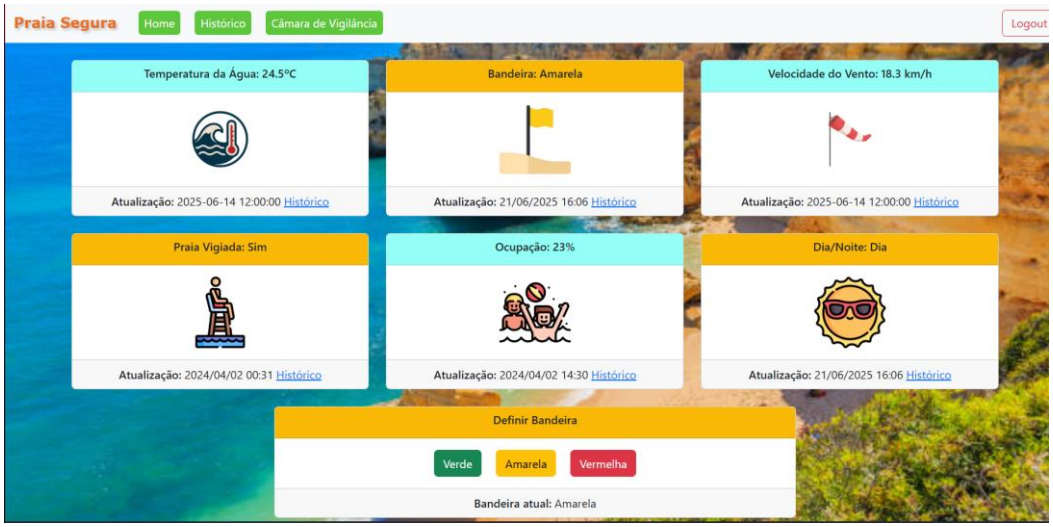

Cenário onde se clica no botão “Verde” :

- No semáforo irá se acender a luz Verde




Cenário onde se clica no botão "Amarela" :

- No semáforo irá se acender a luz amarela



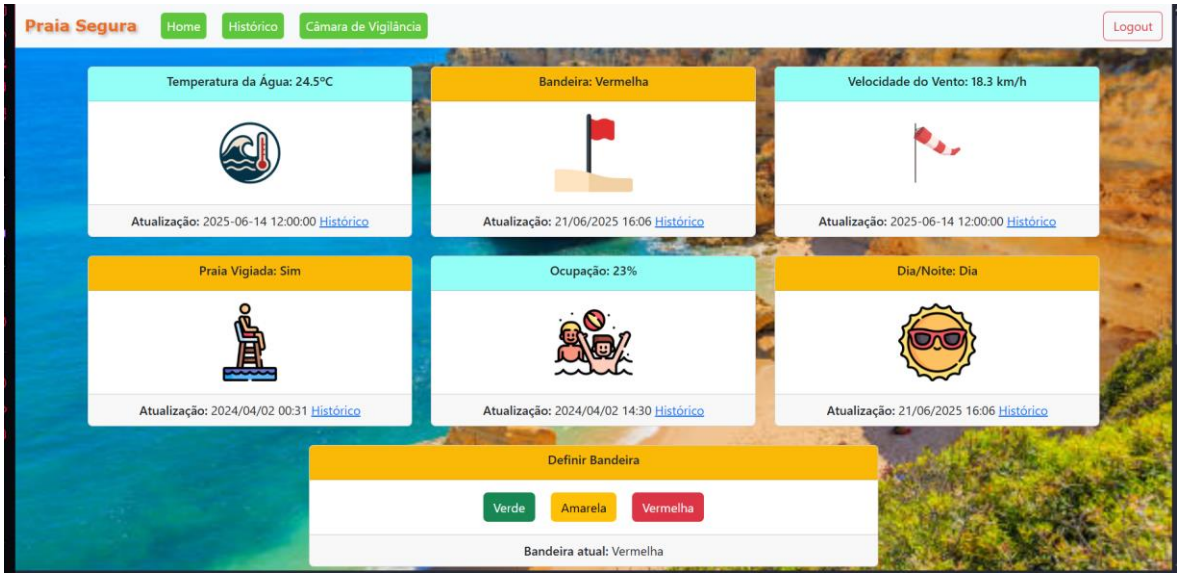

Cenário onde se clica no botão “Vermelha” :

- No semáforo irá se acende a luz Vermelha

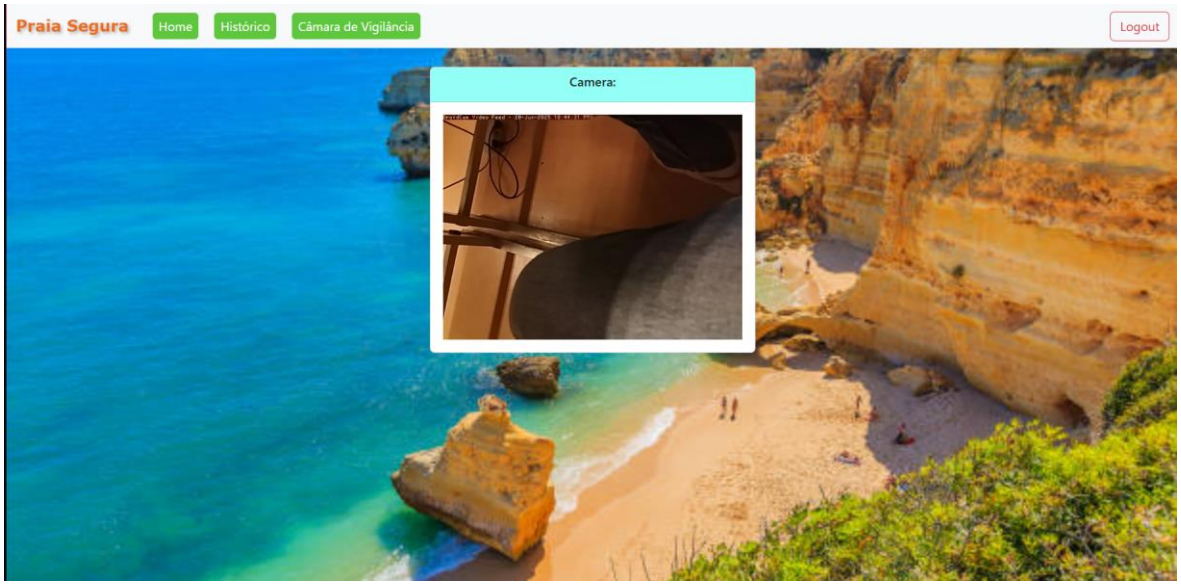
- O Buzzer irá ligar neste cenário

- A Led presente no Rasperry vai acender

- A camara irá tirar uma foto




Exemplo de foto tirada :




Exemplo da tabela com valores aleatórios :

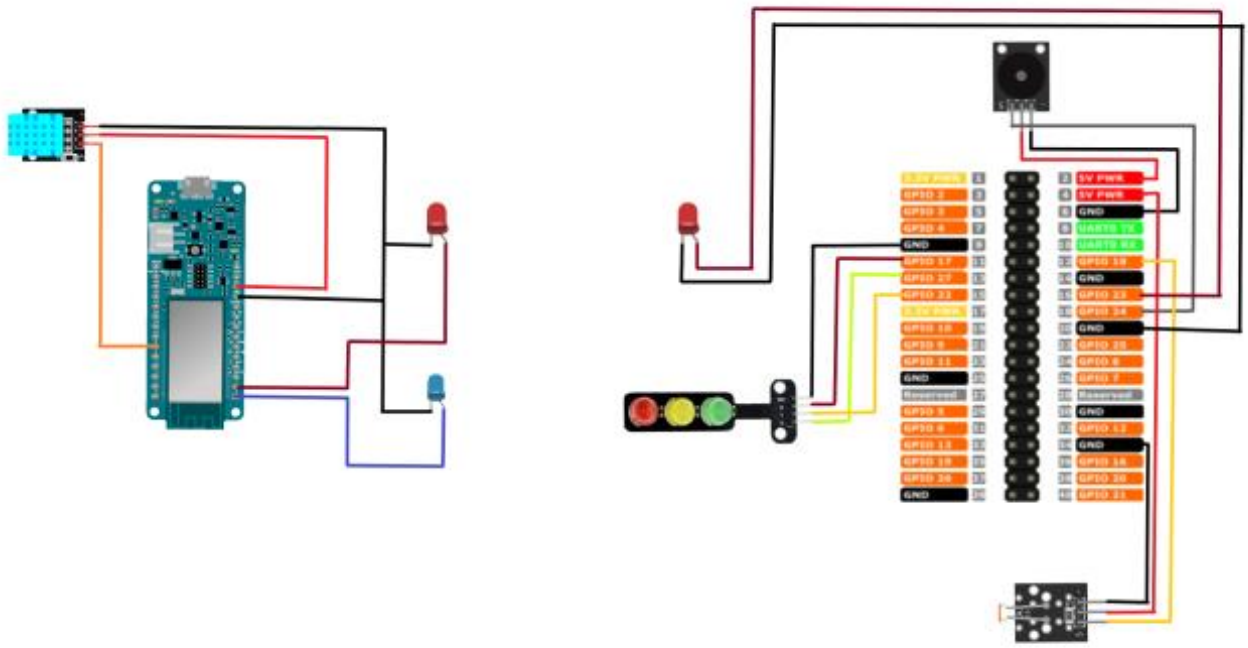
**Informações Úteis**

| Nome | Valor | Ultima Atualização | Estado |
|---|---|---|---|
| **Praia vigiada** | Sim | 2024/04/02 00:31 | Sim |
| **Tempagua** | 24.5ºC | 2025-06-14 12:00:00 | Morna |
| **Vento** | 18.3 km/h | 2025-06-14 12:00:00 | Baixa |
| **Bandeira** | Vermelha | 21/06/2025 16:06 | Vermelha |
| **Ocupação** | 23% | 2024/04/02 14:30 | Baixa |
| **Dia/Noite** | Dia | 21/06/2025 16:06 | Dia |

Cenário completo para este exemplo :

- A temperatura da agua é superior a 20 ou seja o LED da temperatura presente no Rasperry irá acender
- A bandeira está Vermelha logo irá acontecer tudo o que descrevemos no cenario de bandeira vermelha
- O LDR comunicou que estava de dia logo o LED presente no arduino irá estar apagado

## Exemplo da Montagem



# 6. Conclusão

O sistema integrado (Arduíno para sensores e Raspberry Pi para comunicação e imagem) cumpriu os objetivos iniciais: medição de temperatura, humidade e luminosidade, sinalização de "Dia/Noite" e cor de bandeira, acionamento de LEDs e buzzer, e captura/envio de imagens sob condição definida. A modularidade e o uso de requisições HTTP mostraram-se fiáveis, com leituras consistentes e respostas imediatas dos atuadores.

Trabalhar neste projeto foi, por vezes, um verdadeiro desafio. Enfrentámos momentos difíceis devido à escassez de material fornecido e a alguns componentes que chegaram avariados. Ainda assim, essas adversidades fortaleceram o nosso espírito de equipa: improvisámos soluções, partilhámos conhecimentos e mantivemo-nos focados até superar cada obstáculo. No fim, a satisfação de ver todo o sistema a funcionar, desde a leitura de sensores até à captura de imagens, compensou o esforço investido e confirmou que atingimos os objetivos definidos.

# 7. Bibliografia

Para o desenvolvimento do Projeto Praia Inteligente usamos todos os recursos disponibilizados pelos docentes da disciplina de TI, incluindo websites recomendados e aprendizagens retiradas das resoluções das fichas praticas.

OPENAI – ChatGPT-4: ChatGPT

Draw.io: draw.io