**UNIVERSIDADE DO VALE DO ITAJAÍ**
**ESCOLA DO MAR, CIÊNCIA E TECNOLOGIA**
**CURSO DE ENGENHARIA DE COMPUTAÇÃO**
**PROJETO DE SISTEMAS EMBARCADOS**

# MONITOR DE VASOS PARA DECORAÇÃO

por

Guilherme Santana
Thiago Bazilio

Itajaí (SC), novembro de 2025

**UNIVERSIDADE DO VALE DO ITAJAÍ**
**ESCOLA DO MAR, CIÊNCIA E TECNOLOGIA**
**CURSO DE ENGENHARIA DE COMPUTAÇÃO**
**PROJETO DE SISTEMAS EMBARCADOS**

# MONITOR DE VASOS PARA DECORAÇÃO

por

Guilherme Santana
Thiago Bazilio

Relatório apresentado como requisito parcial da disciplina Projeto de Sistemas Embarcados do Curso de Engenharia de Computação para análise e aprovação.
Professores Responsáveis: Paulo Roberto Valim

Felipe Viel

Itajaí (SC), novembro de 2025

# Sumário

***1 INTRODUÇÃO4***
*4*
*4*
*4*
*5*
*7*

***8***
*8*
*8*
*9*
*11*
*12*
*12*
***Erro! Indicador não definido.***
*13*

***14***
*14*
*18*
*19*

***20***

***21***

# 1 INTRODUÇÃO

## 1.1 DEFINIÇÃO DO PROBLEMA

Devido às rotinas corridas do dia a dia, muitas pessoas que cultivam plantas acabam não monitorando a umidade do solo destas, nem temperatura e luminosidade dos vasos e local das plantas. Há então uma necessidade de ferramentas e equipamentos que auxiliem na medição e controle destas métricas.

Além disso, com o crescimento do número de pessoas interessadas no cultivo de jardins e plantas ornamentais como hobby ou elemento decorativo, aumenta também a demanda por tecnologias que facilitem esse cuidado. Tais soluções podem tanto auxiliar os iniciantes no aprendizado quanto tornar mais prática a rotina dos entusiastas mais experientes.

## 1.2 SOLUÇÕES EXISTENTES

As soluções já existentes similares a este projeto consistem em sensores para o solo que medem principalmente a umidade, a temperatura, o PH, a condutividade elétrica e o NPK da planta. Algumas destas soluções, são utilizadas manualmente, sendo necessário colocar o sensor na terra para medição, não possuindo maneira de monitoramento contínuo. Também, um diferencial desta proposta é a possibilidade de visualizar a informação a distância, sem a necessidade de um display no equipamento, como é para diversas soluções já existentes.

## 1.3 SOLUÇÃO PROPOSTA

Para facilitar o cuidado com as plantas, será desenvolvido um equipamento capaz de ler e informar a quantidade de água no solo, a temperatura do ambiente e a luminosidade em que a planta está disposta, buscando mantê-la saudável e auxiliando seu cuidado.

Além disso, o sistema exportara estes dados para um dashboard externo, para que seja possível observar os dados a partir de um dispositivo móvel, a partir de uma rede wi-fi. Desta maneira, o cultivador da planta poderá sempre se manter informado das métricas necessárias para o cuidado de sua planta.

## 1.4 MERCADO

Este produto será projetado para pessoas que possuem o hobby ou que trabalham diretamente na elaboração de vasos para decoração ou no cuidado de pequenas hortas. Em pesquisas, foi apurado que contando apenas com produtores que trabalham diretamente na elaboração de vasos de flores existem 8 mil pessoas, expandindo para 209 mil se contarmos as pessoas que trabalham diretamente com esta prática. Também, apenas em 2021, foi gerado um faturamento maior que 10,9 bilhões de reais, sendo um mercado bastante buscado por pessoas que buscam flores para decoração.

Quadro 1 – Comparativo da solução proposta com as soluções existentes

| **Nome** | **Característca** | **Característc a** | **Característca** | **Característca** |
|---|---|---|---|---|
| Medidor de Ph digital 6 em 1 | Mede até 6 dados diferentes | Possui display digital para visualizar as informações | Deve ser medido manualmente | Funciona a partir de pilhas |
| Medidor De Solo Plantas Digital 3 em 1 PH Umidade Luz Termômetro | Mede até 3 dados diferentes | Possui display analógico para visualizar informações | Deve ser medido manualmente | Funciona a partir de uma bateria |
| Sensor inteligente 5 em 1 com app registrador de dados | Mede até 5 dados diferentes | Utiliza seu celular para visualização dos dados | Deve ser medido manualmente | Funciona conectado a um celular |
| Solução proposta | Mede até 3 dados diferentes | Utiliza um website para visualização dos dados | É instalado uma vez e mede automaticamente | Funciona a partir de um módulo de bateria |


O primeiro produto similar a este projeto é o Medidor de pH digital 6 em 1, que utiliza duas antes com sensores nas pontas para medir 6 dados diferentes, sendo eles pH do solo, fertilidade, temperatura do solo, umidade do solo, umidade do ar e luminosidade ambiente. Estes dados são exibidos a partir de um display digital e deve ser utilizado manualmente para a medição.


Outro produto similar é o Medidor De Solo Plantas Digital 3 em 1 PH Umidade Luz Termômetro que utiliza também de duas hastes de metal com sensores que medem o pH, a humidade e a temperatura do solo, sendo necessário ser instalado e ativado manualmente, exibindo os dados por meio de um display analógico.


Por fim, outro produto similar é o Sensor inteligente 5 em 1 com app registrador de dados que envia os dados de NPK, condutividade elétrica, pH, temperatura e umidade, utilizando 5 hastes com sensores para a medição destes dados. Diferentes das soluções anteriores, esta utiliza seu dispositivo móvel para registrar os dados, sendo necessário pluga-lo para uso, para utiliza-lo como um display de dados e armazenador.

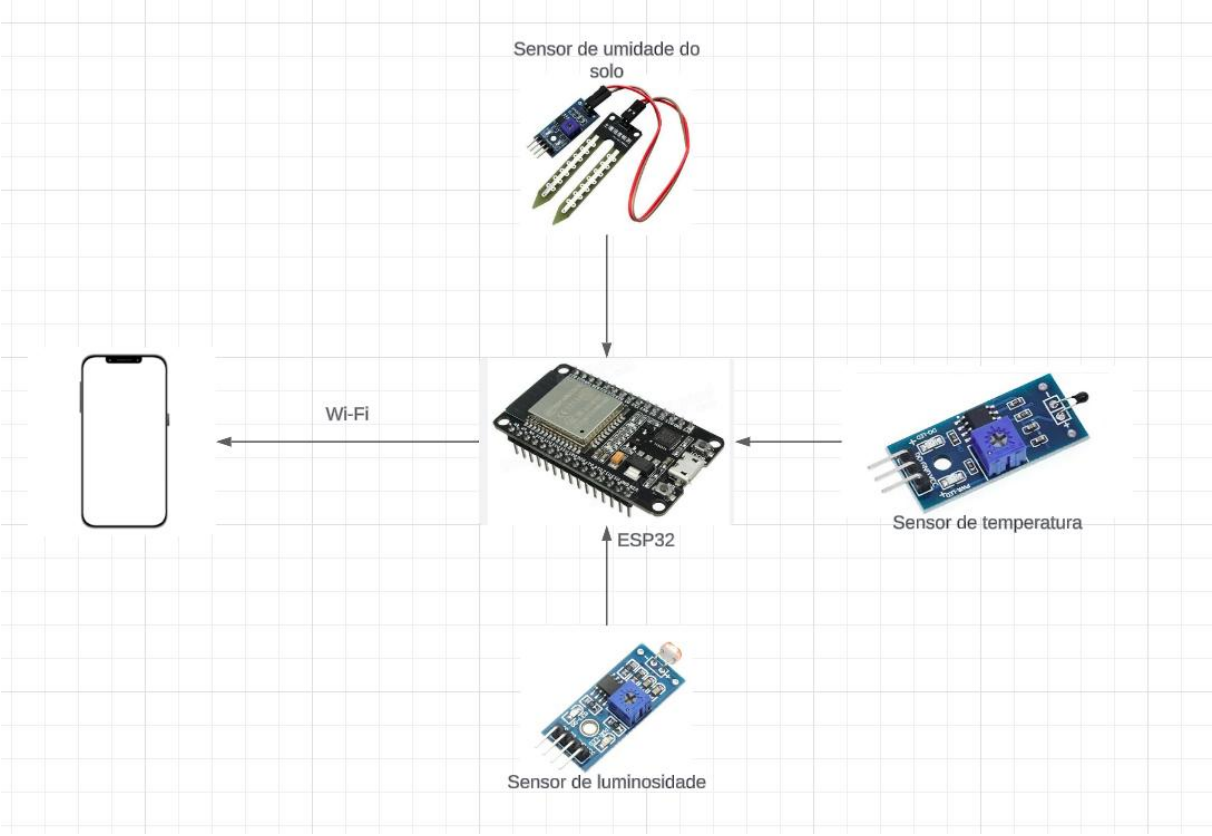
## 1.5 ANÁLISE DE VIABILIDADE

O desenvolvimento do projeto proposto é viável, considerando que há, na grade da matéria, tempo hábil para a integração dos três sensores propostos e para a elaboração de um dashboard para a visualização correta das informações, também, é importante ressaltar que o nível de dificuldade para a elaboração deste projeto não é muito alta, sendo possível finalizar este pojeto no tempo que foi concedido.

# 2 PROJETO

## 2.1 VISÃO GERAL

O sistema utilizará uma ESP32 para integração de 3 sensores e funcionará medindo periodicamente a umidade do solo onde foi instalado, a luminosidade do local e a temperatura do ambiente. Após a coleta dos dados, enviará estas informações via Wi-Fi para uma página web gerada pela própria ESP32 ou enviando estes dados para uma página web que poderá ser acessada pelo navegador. Para energizar o sistema, será utilizado um módulo de bateria que ligará a ESP32 e os sensores para funcionarem de acordo. As medições serão feitas a cada 10 segundos, pensando que qualquer destas métricas estando abaixo do estimulado fazem mal para a planta.




## 2.2 PREMISSAS

- P01: O sistema será apenas um protótipo.
- P02: O sistema utilizará de sensores de baixo custo.

- P03: Assume-se que o local onde o dispositivo será instalado possui acesso mínimo a rede Wi-Fi ou alternativa de comunicação.
- P04: Assume-se que o usuário terá acesso a um dispositivo (smartphone ou computador) para visualizar o dashboard.
- P05: O sistema não ficará ligado por várias horas na realização dos testes.
- P06: Assume-se que as medições ambientais não ultrapassarão limites extremos que possam danificar os sensores.
- P07: Não será realizado testes em ambientes e plantas diversificadas.
- P08: Assume-se que o usuário final terá conhecimento básico para instalação física do dispositivo no vaso.

## 2.3 ANÁLISE DE REQUISITOS

### 2.3.1 Requisitos funcionais

Requisitos Microcontrolador

- RF01: O sistema deve medir periodicamente a umidade do solo da planta.
- RF02: O sistema deve medir a temperatura ambiente próxima ao vaso.
- RF03: O sistema deve medir a luminosidade recebida pela planta.
- RF04: O sistema deve permitir a configuração do nível mínimo das medições.
- RF05: O sistema notificará o usuário caso alguma das medições esteja abaixo do indicado.
- RF06: O sistema deve repetir a medição caso retorne algum valor inválido.
- RF07: O sistema deve armazenar temporariamente as medições coletadas.
- RF08: O sistema deve enviar os dados para um dashboard remoto.
- RF09: O sistema bloqueará o envio para dashboard caso nível de bateria esteja baixo.

Requisitos Dashboard

- RF01: O sistema deve permitir visualização das métricas pelo usuário final.

- RF02: O sistema deve alertar o usuário quando valores estiverem inadequados.
- RF03: O sistema deve operar de forma contínua e sem interrupções.
- RN04: O dashboard deverá ser acessível via navegador ou aplicativo móvel.

### 2.3.2 Requisitos não funcionais

- RNF01: O sistema será prototipado utilizando um ESP32 DevKit.
- RNF02: O código será desenvolvido utilizando linguagem C embarcada.
- RNF03: O sistema deverá utilizar comunicação Wi-Fi.
- RNF04: O protótipo deverá custar no máximo R$300,00.
- RNF05: O protótipo deverá consumir no máximo 0,2 Amperes em operação.
- RNF06: O protótipo deverá ter massa inferior a 500 gramas.
- RNF07: As dimensões somadas não deverão exceder 200 mm totais.
- RNF08: O tempo máximo entre leitura e envio será de 10 segundos.
- RNF09: O sistema deverá realizar pelo menos uma amostra a cada 10 segundos.

### 2.3.3 Regras de negócio

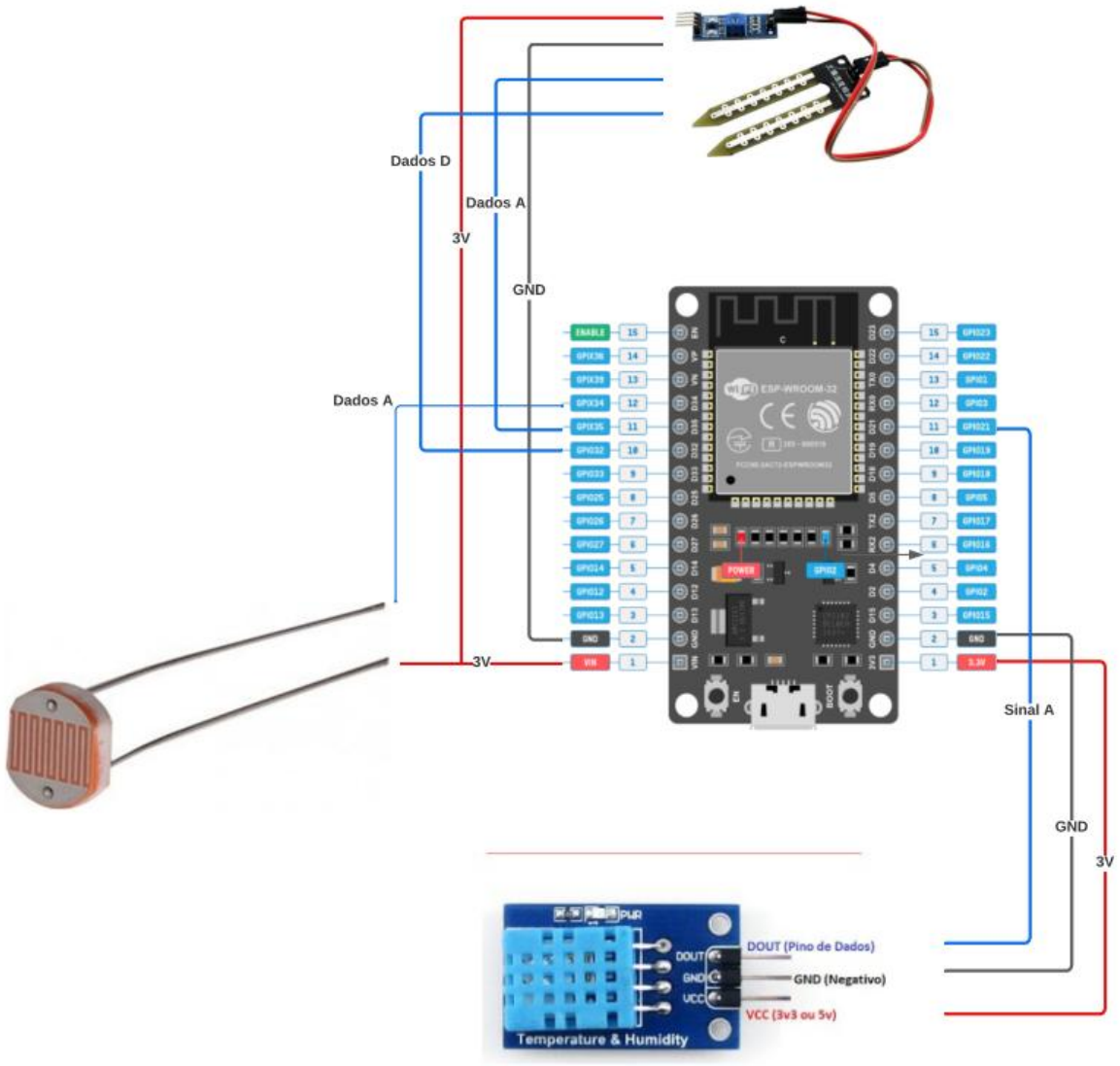
Nesta seção, apresente as regras de negócio do sistema a ser desenvolvido.

- RN01: Se a umidade estiver abaixo do limite mínimo, um alerta deverá ser gerado ao usuário.
- RN02: Se a temperatura ultrapassar o limite definido, o sistema deverá registrar o evento como crítico.
- RN03: Se a luminosidade ficar insuficiente por longos períodos, o dashboard deverá notificar o usuário.
- RN04: Se qualquer sensor retornar valores inválidos, o sistema deverá descartar a leitura e repetir a medição.
- RN05: O sistema deverá bloquear o envio de dados caso a bateria esteja abaixo do nível seguro.

- RN06: Alertas simultâneos deverão ser priorizados pela gravidade da métrica (temperatura > umidade > luminosidade).

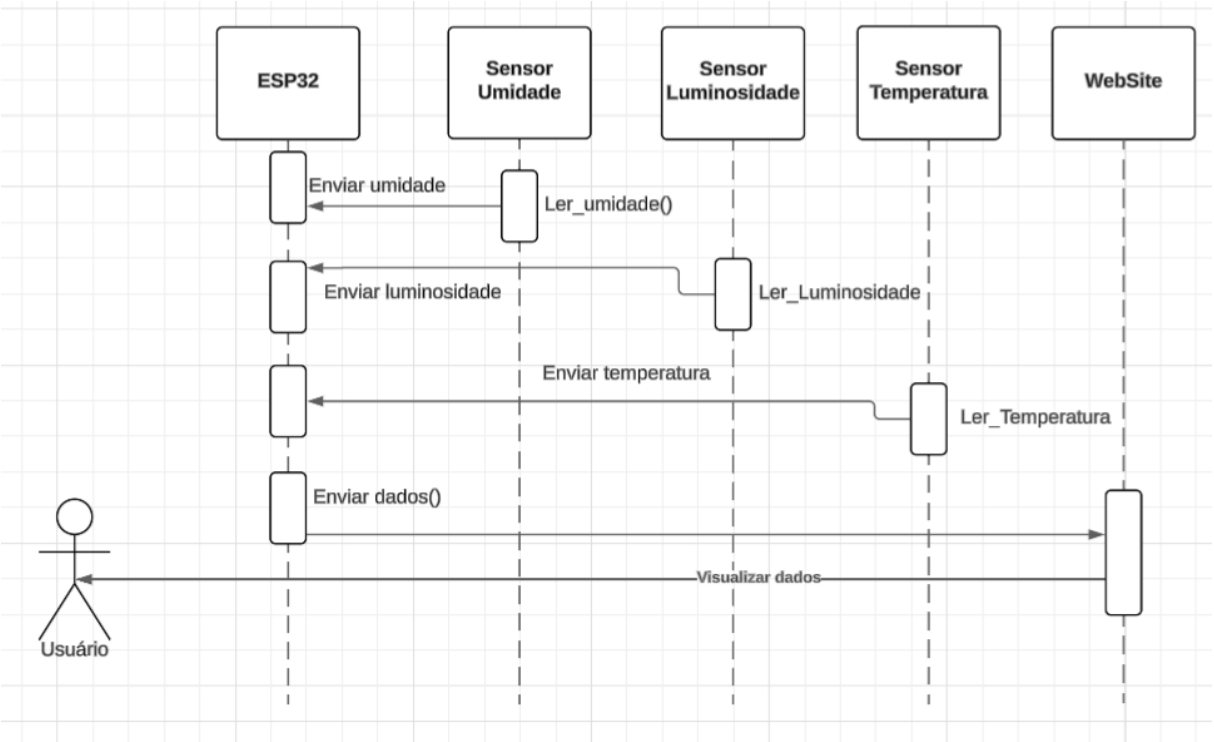
## 2.4 ARQUITETURA DE HARDWARE

Na parte de arquitetura de hardware, será conectado os 3 sensores na porta que fornece energia de 3 volts, para manter eles funcionando, o fio de ground, para fechar o circuito e a entrada analógica que enviará os dados dos sensores para a ESP32.



## 2.5 ARQUITETURA DE SOFTWARE

Nesta seção, demostramos o funcionamento do software, onde a ESP32 receberá os valores dos sensores e enviará para um Website gerado pela própria ESP32, possibilitando o usuário a visualizar os resultados lidos pelos sensores.




## 2.6 LISTA DE MATERIAIS E ORÇAMENTO

Apresente uma listagem dos recursos necessários para o desenvolvimento do protótipo, identificação dos seus custos e da real disponibilidade para construção do protótipo, conforme quadro exemplo a seguir. É preciso indicar quem irá adquirir itens que não estejam disponíveis e informar o prazo para recebimento caso o aluno já tenha encomendado.

Quadro 2 – Lista de materiais

| Item | Qtd | Disponível | Responsável(is) |
|---|---|---|---|
| Kit de desenvolvimento ESP32 | 01 | Sim | Alunos |
| Sensor de luminosidade | 01 | Sim | Professores |
| Sensor de temperatura | 01 | Sim | Professores |
| Sensor de umidade | 01 | Sim | Professores |

Quadro 3 – Orçamento (apenas para itens não disponíveis)

| Item | Custo unitário | Qtd | Custo parcial | Fornecedor de referência |
|---|---|---|---|---|
| Kit de desenvolvimento ESP32 | R$ 50,00 | 01 | R$ 50,00 | Mercado livre |
| Sensor de luminosidade | R$ 5,00 | 01 | R$ 5,00 | Mercado livre |
| Sensor de temperatura | R$ 10,00 | 01 | R$ 10,00 | Mercado livre |
| Sensor de umidade | R$ 15,00 | 01 | R$ 15,00 | Mercado livre |

## 2.7 CRONOGRAMA

Quadro 4 – Cronograma de execução

| Atividade | Sem. 1 | Sem. 2 | Sem. 3 | Sem. 4 | Entregável |
|---|---|---|---|---|---|
| Projeto | X | | | | Relatório parcial |
| Implementação | | X | X | X | Projeto parcial |
| Verificação | | | X | X | |
| Avaliação | | | | X | Projeto final |
| Documentação | X | X | X | X | Relatório completo |

# 3 DESENVOLVIMENTO

## 3.1 IMPLEMENTAÇÃO

```c
static const char *TAG = "SENSOR";

#define CANAL_UMIDADE_ADC ADC1_CHANNEL_7 // GPIO35 -> ADC1_CHANNEL_7
#define BRUTO_UMIDADE_SECO 4095 // seco
#define BRUTO_UMIDADE_MOLHADO 1700 // submerso
#define PINO_DHT 21 // Sensor de temperatura (assumido DHT11 HW-036) no GPIO21
#define ADC_MAXIMO 4095.0f
#define VREF 3.3f
#define DHT_MAX_WAIT 1000

#define CANAL_LUZ_ADC ADC1_CHANNEL_4
#define PINO_LUZ 32 // ADC1_CHANNEL_4 -> GPIO32

// WiFi AP
#define WIFI_SSID "thiago_e_guilherme"
#define WIFI_PASS "12345678"

// histórico
#define HIST_TAMANHO 10
#define DELTA_MAX_TEMPERATURA 15

typedef struct {
    int bruto_umidade;
    float pct_umidade;
    int bruto_luz;
    float pct_luz;
    int temperatura;
    int umidade_ar;
    uint32_t ts_ms;
} leitura_t;

static leitura_t historico[HIST_TAMANHO];
static int hist_idx = 0;
static int hist_cnt = 0;
static leitura_t leitura_atual;
static SemaphoreHandle_t mutex_leituras = NULL;
static httpd_handle_t servidor_http = NULL;
static int ultima_temperatura = 0;
static int ultima_umidade_ar = 0;
```

Começamos pela definição das entradas dos sensores, utilizando a porta 32 para o sensor de luminosidade, o 35 para o sensor de umidade e o 21 para o sensor de temperatura. Junto disso, é definido a senha da rede que será gerada pela ESP32 e dos valores padrões que serão utilizados pelo sistema, como por exemplo, os valores de referencia de voltagem e o máximo de registros que serão mostrados na tela.

```c
static bool ler_temperatura(int gpio, int *out_temp, int *out_umi) {
    uint8_t bits[5] = {0};

    gpio_set_direction(gpio, GPIO_MODE_OUTPUT);
    gpio_set_level(gpio, 0);
    vTaskDelay(pdMS_TO_TICKS(20));


    gpio_set_level(gpio, 1);
    wait(30);
    gpio_set_direction(gpio, GPIO_MODE_INPUT);
    gpio_set_pull_mode(gpio, GPIO_PULLUP_ONLY);

    // aguarda resposta do sensor
    int64_t tstart = esp_timer_get_time();
    while (gpio_get_level(gpio) == 1) {
        if ((esp_timer_get_time() - tstart) > DHT_MAX_WAIT * 100) return false;
    }

    tstart = esp_timer_get_time();
    while (gpio_get_level(gpio) == 0) {
        if ((esp_timer_get_time() - tstart) > DHT_MAX_WAIT * 100) return false;
    }
    tstart = esp_timer_get_time();
    while (gpio_get_level(gpio) == 1) {
        if ((esp_timer_get_time() - tstart) > DHT_MAX_WAIT * 100) return false;
    }

    for (int i = 0; i < 40; i++) {
        tstart = esp_timer_get_time();
        while (gpio_get_level(gpio) == 0) {
            if ((esp_timer_get_time() - tstart) > DHT_MAX_WAIT * 100) return false;
        }
        int64_t t0 = esp_timer_get_time();
        while (gpio_get_level(gpio) == 1) {
            if ((esp_timer_get_time() - t0) > DHT_MAX_WAIT * 100) break;
        }
    for (int i = 0; i < 40; i++) {
        tstart = esp_timer_get_time();
        while (gpio_get_level(gpio) == 0) {
            if ((esp_timer_get_time() - tstart) > DHT_MAX_WAIT * 100) return false;
        }
        int64_t t0 = esp_timer_get_time();
        while (gpio_get_level(gpio) == 1) {
            if ((esp_timer_get_time() - t0) > DHT_MAX_WAIT * 100) break;
        }
        int pulse_len = (int)(esp_timer_get_time() - t0);
        int byte_idx = i / 8;
        bits[byte_idx] <<= 1;
        if (pulse_len > 50) bits[byte_idx] |= 1;
    }

    // soma
    uint8_t checksum = bits[0] + bits[1] + bits[2] + bits[3];
    if (checksum != bits[4]) return false;

    *out_umi = 100 - bits[0];
    *out_temp = bits[2];
    return true;
}
```

Nesta função, realizamos a leitura da temperatura, coletando os dados necessário para ser posteriormente mostrado na tela do Website gerado pela ESP32.

```c
static void tarefa_sensores(void *arg) {
    adc1_config_width(ADC_WIDTH_BIT_12);
    adc1_config_channel_atten(CANAL_UMIDADE_ADC, ADC_ATTEN_DB_11);
    adc1_config_channel_atten(CANAL_LUZ_ADC, ADC_ATTEN_DB_11);

    gpio_set_direction(PINO_LUZ, GPIO_MODE_INPUT);
    gpio_set_pull_mode(PINO_LUZ, GPIO_PULLDOWN_ONLY);

    while (1) {
        const int samples = 8;
        int bruto_umidade = 0;
        float pct_umidade = 0.0f;
        ler_umidade_solo(samples, &bruto_umidade, &pct_umidade);
        ESP_LOGI(TAG, "Umidade do solo: %.1f%%", pct_umidade);

        int bruto_luz = 0;
        float pct_luz = 0.0f;
        ler_luminosidade(samples, &bruto_luz, &pct_luz);
        // ESP_LOGI(ETIQUETA, "Sensor de Luminosidade - bruto: %d  Brilho: %.1f%%", bruto_luz, pct_luz);
        ESP_LOGI(TAG, "Brilho: %.1f%%", pct_luz);

int t = ultima_temperatura;
int h = ultima_umidade_ar;
bool ok = false;
// tentar até 3 vezes ler, em caso de falha manter último valor válido
for (int tentativa = 0; tentativa < 3; tentativa++) {
    if (ler_temperatura(PINO_DHT, &t, &h)) {
        ok = true;
        break;
    }
    vTaskDelay(pdMS_TO_TICKS(200));
}
if (ok) {
    // se já tivermos um último valor válido, rejeitar saltos muito grandes
    //fix: nos casos de salto muito grande (>15) a temperatura é exatamente o dobro do val
    if (ultima_temperatura != 0 && abs(t - ultima_temperatura) > DELTA_MAX_TEMPERATURA) {
        int metade = t / 2;
        if (abs(metade - ultima_temperatura) <= DELTA_MAX_TEMPERATURA) {
            t = metade;
            ultima_temperatura = t;
            ultima_umidade_ar = h;
        } else {
            t = ultima_temperatura;
            h = ultima_umidade_ar;
        }
    } else {
        ultima_temperatura = t;
        ultima_umidade_ar = h;
    }
} else {
    t = ultima_temperatura;
    h = ultima_umidade_ar;
}
ESP_LOGI(TAG, "Temperatura: %dC", t);
```

Nesta etapa do código é onde realizamos as chamadas dos sensores, para que realizem suas leituras e retornem para a ESP. Caso alguma leitura seja feita de maneira incorreta, utilizamos a última leitura correta para amostragem no website.

```
// atualizar leitura atual e histórico
leitura_t nova;
nova.bruto_umidade = bruto_umidade;
nova.pct_umidade = pct_umidade;
nova.bruto_luz = bruto_luz;
nova.pct_luz = pct_luz;
nova.temperatura = t;
nova.umidade_ar = h;
nova.ts_ms = (uint32_t)(esp_timer_get_time() / 1000);

if (mutex_leituras) {
    if (xSemaphoreTake(mutex_leituras, pdMS_TO_TICKS(200))) {
        leitura_atual = nova;
        // inserir no histórico circular
        historico[hist_idx] = nova;
        hist_idx = (hist_idx + 1) % HIST_TAMANHO;
        if (hist_cnt < HIST_TAMANHO) hist_cnt++;
        xSemaphoreGive(mutex_leituras);
    }
}

ESP_LOGI(TAG, "-------------------------------------");
vTaskDelay(pdMS_TO_TICKS(3000));
```

Nesta parte, foi onde montamos o histórico das informações medidas anteriormente, para que seja possível observar a alteração dos dados medidos.

```
// página HTML
static esp_err_t index_get_handler(httpd_req_t *req) {
    const char* pagina =
        "<!doctype html><html><head><meta charset=\"utf-8\">"
        "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">"
        "<title>Leituras</title></head><body><h1>Leituras sensores</h1>"
        "<div><strong>Atual</strong><div id=\"atual\">carregando...</div></div>"
        "<h2>Ultimas 10 leituras</h2><div id=\"historico\">carregando...</div>"
        "<script>"
        "function fmt(v){return v==null?'-':v}"
        "function atualizar(){"
        "  fetch('/values').then(r=>r.json()).then(j=>{"
        "    const a = j.atual;"
        "    document.getElementById('atual').innerHTML = 'Umidade solo: ' + a.pct_umidade.toFixed(1) + '% <br> Luminosidade: '
        "    let html = '';"
        "    j.historico.forEach(function(h){"
        "      var d = new Date(h.ts_ms);"
        "      html += '<div>' + d.toLocaleTimeString() + ':  Umidade do solo: ' + h.pct_umidade.toFixed(1) + '%  Luminosidade:
        "    });"
        "    document.getElementById('historico').innerHTML = html;"
        "  });"
        "}"
        "atualizar();setInterval(atualizar,3000);"
        "</script></body></html>";
    httpd_resp_set_type(req, "text/html; charset=utf-8");
    httpd_resp_send(req, pagina, HTTPD_RESP_USE_STRLEN);
    return ESP_OK;
```

Nesta etapa produzimos o HTML que será gerado pela ESP para amostragem dos dados, utilizando os valores medidos pelos sensores para ser observado pelo usuário.

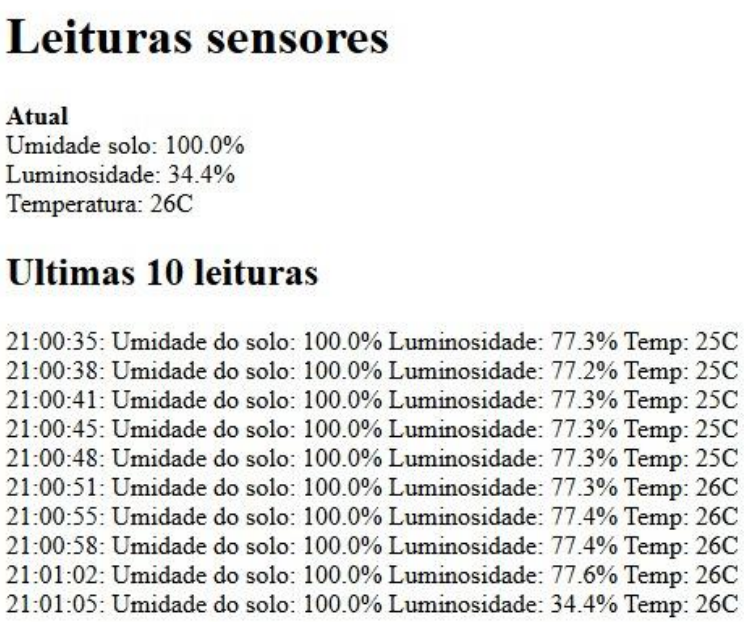
```
        ESP_LOGI(TAG, "Servidor HTTP iniciado");
    } else {
        ESP_LOGE(TAG, "Falha ao iniciar servidor HTTP");
    }

    xTaskCreate(tarefa_sensores, "tarefa_sensores", 4096, NULL, 5, NULL);
}
```

Por fim, levantamos a página web para ser acessada pelo usuário.

## 3.2 VERIFICAÇÃO

Feito testes a partir dos sensores, medindo valores em locais frios e quentes, com pouca ou muita iluminação e em utilizando o sensor de umidade para validar alteração ao colocá-lo em contato direto com a água.

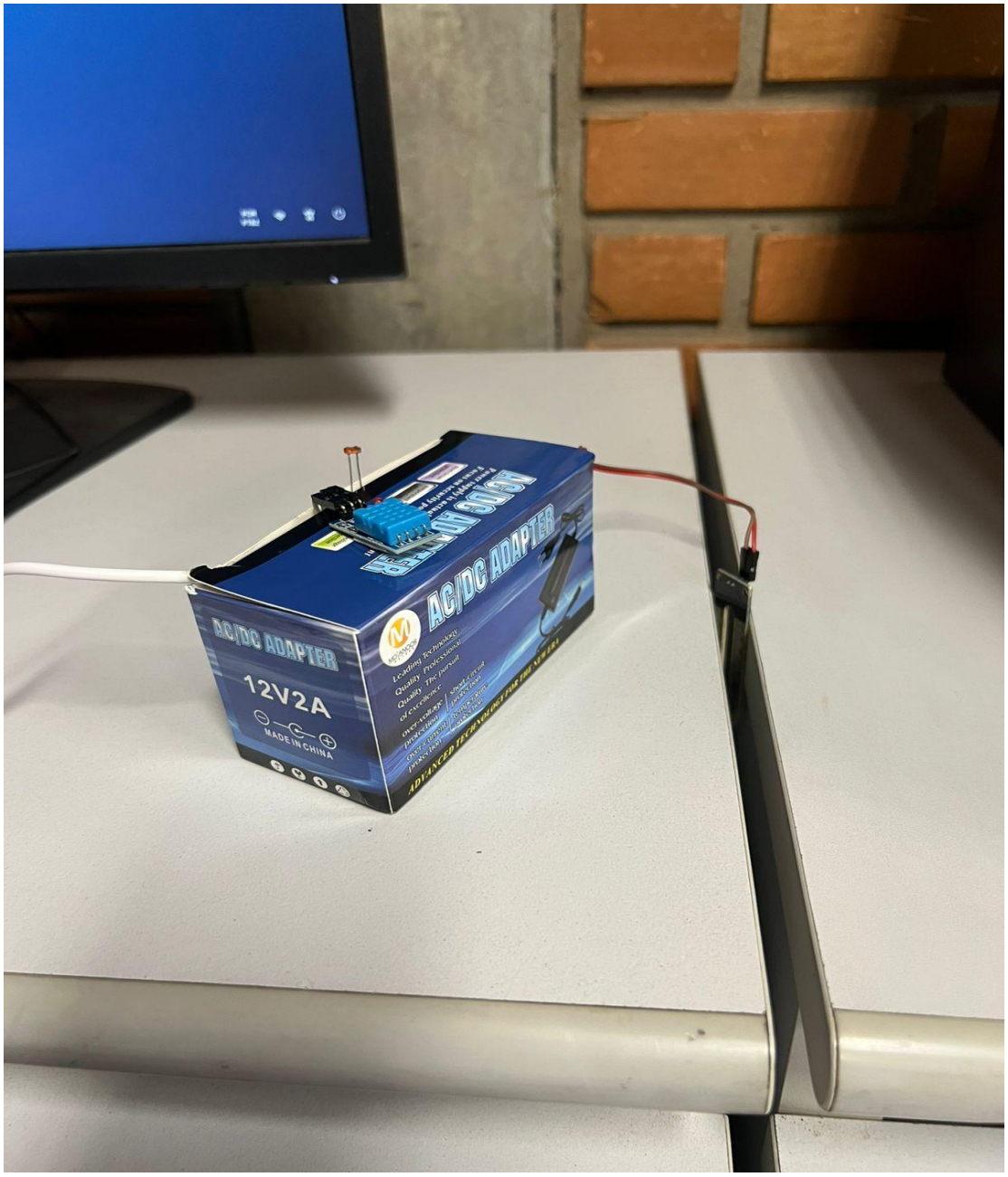
## 3.3 RESULTADOS




Como resultados, obtivemos estas leituras via WEB, mostrando os valores medidos pelos sensores e mantendo as últimas medidas para observar a alteração que teve nos resultados anteriores.

# 4 CONSIDERAÇÕES FINAIS / CONCLUSÕES

Concluímos com este projeto, que é possível montar um projeto para monitoramento de plantas de maneira simples, podendo receber atualizações, como adição de bombas d'água para automatizar ainda mais o projeto, ou conectando o projeto a outras soluções inteligentes, como um termostato ou uma lâmpada inteligente, capazes de mudar as medidas sem interação humana.



# REFERÊNCIAS

https://cnabrasil.org.br/noticias/mercado-de-flores-no-brasil-atingiu-r-10-9-bilhoes-em-2021