CONTROLE E SIMULAÇÃO DE REFRIGERAÇÃO EM CAIXAS DE VACINA COM ESP32

Fernando Lucas Santos Cordeiro

Guskov da Silva Coelho

João Manoel de Souza Muniz

Marcos Corrêa Nunes

Marcos Vinicios da Silva Lima

Paulo Gabriel Sodré Meneses

RESUMO: Este artigo visa evidenciar a criação e funcionamento de um sistema de refrigeração automatizado utilizando a placa microcontroladora ESP32, em conjunto com o sensor de temperatura DHT-11, para controle de ventoinhas que conduzem a corrente de refrigeração de um ambiente para outro. A temperatura é visualizada e monitorada através de um display OLED, e todo o sistema é controlado remotamente pela plataforma Blynk.

Palavra-Chave: Refrigeração, ESP32, Temperatura, Blynk**.**

ABSTRACT: This article aims to highlight the creation and operation of a refrigeration system that uses the ESP32 microcontroller board and works together with the DHT-11 temperature sensor to control fans that direct the cooling current from one environment to another. The temperature is visualized and monitored through an OLED display, and the entire system is remotely controlled via the Blynk platform.

Keywords: Refrigeration, ESP32, Temperature, Blynk.

---

[1] Artigo elaborado para a disciplina Instrumentação Eletrônica do Curso de Engenharia da Computação do Centro Universitário do Maranhão como requisito para obtenção de nota sob a orientação do prof. Madson Cruz Machado.

[2] Discentes do 6º e 7º período de Engenharia da Computação do Centro Universitário do Maranhão (Uniceuma).

1. INTRODUÇÃO

Diante do cenário atual, torna-se cada vez maior a presença da automação em sistemas de controle de temperatura, principalmente para aplicações que exigem um alto grau de precisão e eficiência energética. Dessa forma, foi desenvolvido um projeto inovador de um sistema de refrigeração automático que utiliza a placa microcontroladora ESP32 para o controle da temperatura de um ambiente. Para o acionamento das ventoinhas, o sistema é integrado com o sensor de temperatura DHT11, com o qual a temperatura interna do ambiente é monitorada continuamente. Quando a temperatura medida atinge um valor predefinido, as ventoinhas são automaticamente acionadas para resfriar o ambiente interno, garantindo a manutenção de condições ideais de conservação.

Além do controle automatizado, o sistema terá a opção de controle remoto através da plataforma Blynk. O sistema permite que os usuários monitorem e ajustem a temperatura ideal em tempo real de qualquer lugar que tenha acesso à internet, proporcionando assim, um melhor controle e usabilidade do sistema. Desta forma, ao longo deste artigo, será discutido em detalhes o desenvolvimento e funcionalidade do sistema, desde a ideia inicial até a implementação final. Isso incluirá os componentes, a coordenação de hardware e software e os testes reais para determinar a eficácia do sistema.

1. METODOLOGIA

A metodologia adotada para o desenvolvimento do sistema de refrigeração automatizado utilizando a placa microcontroladora ESP32 foi estruturada em várias etapas, desde o planejamento e a concepção do projeto até a implementação e os testes finais.

1.1 Planejamento e Conceito do Projeto:

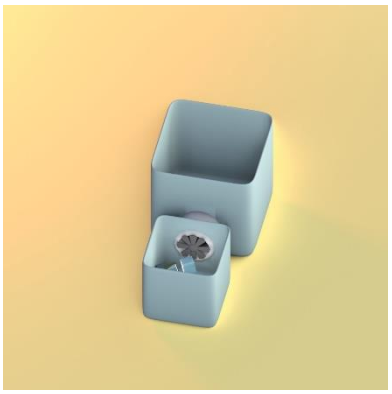
O projeto começou com uma extensa revisão da literatura para selecionar os componentes mais adequados. A ESP32 foi escolhida por suas características robustas, incluindo capacidade de processamento, conectividade e suporte para múltiplas interfaces de sensores. O sensor DHT-11 foi selecionado pela sua simplicidade, precisão e custo-benefício. A plataforma Blynk foi escolhida pela sua interface intuitiva e robustez em aplicações IoT, permitindo controle remoto eficiente e confiável.

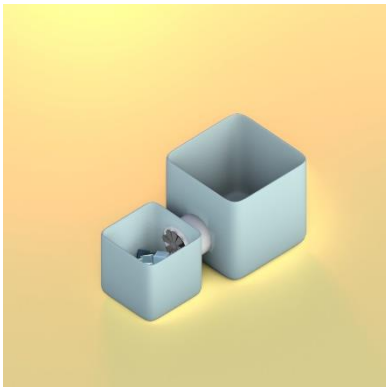
1.2 Design do Sistema:

O design do sistema foi criado utilizando software de modelagem 3D, o que facilitou a disposição dos componentes e a montagem do protótipo. O sistema inclui a microcontroladora ESP32, o sensor DHT-11, ventoinhas para resfriamento, um

display OLED para visualização da temperatura e a plataforma Blynk para controle remoto. O design modular permite expansões futuras e facilita a substituição de componentes conforme necessário.

Figura 1 e 2– Modelo 3D do projeto





1.3 Integração de Hardware

O design do sistema foi criado utilizando software de modelagem 3D, o que facilitou a disposição dos componentes e a montagem do protótipo. O sistema inclui a microcontroladora ESP32, o sensor DHT-11, ventoinhas para resfriamento, um display OLED para visualização da temperatura e a plataforma Blynk para controle remoto. O design modular permite expansões futuras e facilita a substituição de componentes conforme necessário.

1.4 Desenvolvimento de Software

O software foi desenvolvido na IDE do Arduino, utilizando bibliotecas específicas para cada componente. O código foi modularizado para facilitar a leitura do sensor, controle das ventoinhas, interface com o display e conectividade com a Blynk. A seguir, apresenta-se um trecho do código desenvolvido para o controle das ventoinhas e monitoramento de temperatura.

1.4 Explicações de Trechos Importantes do Código

- **Definição e Inicialização do Sensor de Temperatura DHT11:**

O código começa definindo o pino de conexão do sensor DHT11 e o tipo de sensor utilizado. O sensor DHT11 é escolhido pela sua simplicidade e confiabilidade para monitorar a temperatura e a umidade. A definição do pino é feita através da constante DHTPIN, que está configurada para o pino 4 do ESP32. Em seguida, o tipo do sensor é especificado pela macro DHTTYPE, definida como DHT11. A função dht.begin() é chamada para inicializar o sensor, garantindo que ele esteja pronto para realizar leituras precisas de temperatura. Essa configuração é fundamental para garantir que o sistema possa monitorar continuamente a temperatura ambiente e reagir conforme necessário para manter a temperatura desejada.

- **Configuração do Display OLED:**

O display OLED é configurado com um endereço I2C padrão (0x3c) e dimensões específicas de 128x64 pixels, que são definidas através das macros SCREEN_WIDTH e SCREEN_HEIGHT. O display é utilizado para exibir a temperatura monitorada em tempo real, proporcionando uma visualização clara e imediata dos dados. A inicialização do display é realizada pela função display.begin(), que verifica se o display está corretamente conectado e funcionando. Em caso de falha, uma mensagem de erro é exibida no monitor serial, e o código entra em um loop infinito para evitar o funcionamento incorreto. A configuração do display é essencial para a interface do usuário, permitindo que os dados de temperatura sejam facilmente acessados e monitorados.

- **Controle Automático das Ventoinhas:**

O sistema utiliza duas ventoinhas, cujos estados são controlados pelos pinos 16 e 17, configurados como saídas através da função pinMode. O estado das ventoinhas é armazenado em variáveis booleanas (fanPinState e fanPinTwoState), que indicam se as ventoinhas estão ligadas ou desligadas. A função fanController() verifica a temperatura atual lida pelo sensor DHT11 e decide se as ventoinhas devem ser ativadas ou desativadas. Se a temperatura estiver abaixo de 30°C, as ventoinhas são desligadas usando a função fanOff(), e caso contrário, são ligadas com a função fanOn(). Este controle automático garante que a temperatura do ambiente seja mantida dentro dos limites desejados, promovendo uma operação eficiente e minimizando o consumo de energia. A função de controle das ventoinhas é crítica para o funcionamento do sistema de refrigeração automatizado, pois ajusta a ventilação com base nas condições de temperatura monitoradas em tempo real.

```
// Definindo pinos e tipo do sensor DHT
const int DHTPIN = 4;  // Pino onde o sensor DHT11 está conectado
#define DHTTYPE DHT11  // Tipo do sensor DHT utilizado
DHT dht(DHTPIN, DHTTYPE);  // Inicializando o sensor DHT

// Endereço I2C do display OLED
#define i2c_Address 0x3c  // Endereço padrão I2C do display
#define SCREEN_WIDTH 128  // Largura do display
#define SCREEN_HEIGHT 64  // Altura do display
#define OLED_RESET -1     // Não é usado o pino de reset
Adafruit_SH1106G display = Adafruit_SH1106G(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);  // Inicializando o display

// Pino das Ventoinhas
const int fanPin = 16;  // Pino para a primeira ventoinha
const int fanPinTwo = 17;  // Pino para a segunda ventoinha
bool fanPinState = LOW;  // Estado inicial da primeira ventoinha (desligada)
bool fanPinTwoState = LOW;  // Estado inicial da segunda ventoinha (desligada)

void setup() {
    // Inicializa a comunicação serial
    Serial.begin(9600);
    // Inicializa o Blynk
    Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
    // Inicializa o sensor DHT
    dht.begin();
    // Inicializa o display com o endereço I2C correto
    if (!display.begin(i2c_Address, true)) {
        Serial.println(F("Falha na alocação do SSD1306"));
        while (true); // Não prossegue, fica em loop infinito
    }
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SH110X_WHITE);
    // Configura os pinos das ventoinhas como saídas
    pinMode(fanPin, OUTPUT);
    pinMode(fanPinTwo, OUTPUT);
}

// Função para obter a temperatura
float getTemperature() {
    return dht.readTemperature();  // Retorna a temperatura lida pelo sensor DHT11
}
```

```
// Função para atualizar o display LCD
void updateDisplay() {
    delay(100); // Espera o OLED ligar
    display.clearDisplay();
    display.setCursor(0, 10);
    float temperature = getTemperature();
    display.print(F("Temperatura: "));
    display.print(temperature);
    display.println(F(" C"));
    Blynk.virtualWrite(V4, temperature);
    display.display();
}

// Função controladora da ventoinha
void fanController() {
    if (getTemperature() <= 30) {
        fanOn();  // Liga as ventoinhas se a temperatura estiver abaixo de 30°C
    } else {
        fanOff();  // Desliga as ventoinhas se a temperatura estiver acima de 30°C
    }
}

// Liga a ventoinha (GPIO HIGH)
void fanOn() {
    digitalWrite(fanPin, HIGH);  // Liga a primeira ventoinha
    digitalWrite(fanPinTwo, HIGH);  // Liga a segunda ventoinha
}

// Desliga a ventoinha (GPIO LOW)
void fanOff() {
    digitalWrite(fanPin, LOW);  // Desliga a primeira ventoinha
    digitalWrite(fanPinTwo, LOW);  // Desliga a segunda ventoinha
}

void loop() {
    Blynk.run();  // Mantém a conexão com a plataforma Blynk
    updateDisplay();
    if (fanPinState == LOW && fanPinTwoState == LOW) {
        fanController();  // Controla as ventoinhas com base na temperatura
    }
}
```

- **Comunicação com a Plataforma Blynk:**

O código utiliza a plataforma Blynk para permitir o controle remoto e o monitoramento em tempo real do sistema de refrigeração. A função Blynk.begin(auth, ssid, pass, "blynk.cloud", 80) inicializa a comunicação com o servidor Blynk, utilizando as credenciais de autenticação (auth), o nome da rede Wi-Fi (ssid) e a senha (pass). Essa integração é crucial para possibilitar que os usuários monitorem a temperatura e controlem as ventoinhas através de um aplicativo móvel ou da interface web do Blynk. Os comandos BLYNK_WRITE(V16) e BLYNK_WRITE(V17) são usados para receber informações do aplicativo Blynk, permitindo que os usuários ativem ou desativem as ventoinhas remotamente. Isso adiciona uma camada de flexibilidade e conveniência ao sistema, permitindo ajustes rápidos e precisos sem a necessidade de acesso físico ao dispositivo.

- **Loop Principal (loop()):**

O loop principal do código é responsável por manter o funcionamento contínuo do sistema de refrigeração automatizado. Dentro deste loop, Blynk.run() é chamado para manter a conexão com o servidor Blynk e garantir a recepção e envio contínuo de dados. A função updateDisplay() atualiza regularmente o display OLED com a temperatura medida pelo sensor DHT11, permitindo que os usuários visualizem a temperatura atualizada em tempo real. A função fanController() verifica constantemente a temperatura ambiente e controla o estado das ventoinhas com base nos valores predefinidos. Se a temperatura exceder o limite configurado (30°C neste caso), fanOff() é chamada para desligar as ventoinhas e economizar energia; caso contrário, fanOn() é chamada para ligar as ventoinhas e resfriar o ambiente conforme

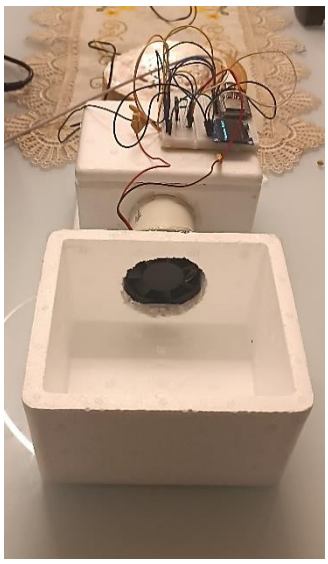
necessário. Este loop contínuo garante que o sistema opere de forma eficiente e responsiva, mantendo a estabilidade térmica do ambiente monitorado.

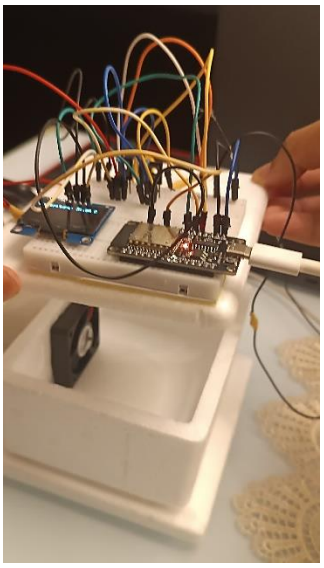
```
void setup() {
    // Inicializa a comunicação serial
    Serial.begin(9600);
    // Inicializa o Blynk
    Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
    // Inicializa o sensor DHT
    dht.begin();
    // Inicializa o display com o endereço I2C correto
    if (!display.begin(i2c_Address, true)) {
        Serial.println(F("Falha na alocação do SSD1306"));
        while (true); // Não prossegue, fica em loop infinito
    }
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SH110X_WHITE);
    // Configura os pinos das ventoinhas como saídas
    pinMode(fanPin, OUTPUT);
    pinMode(fanPinTwo, OUTPUT);
}

// Função controladora da ventoinha
void fanController() {
    if (getTemperature() <= 30) {
        fanOn();  // Liga as ventoinhas se a temperatura estiver abaixo de 30°C
    } else {
        fanOff();  // Desliga as ventoinhas se a temperatura estiver acima de 30°C
    }
}

// Liga a ventoinha (GPIO HIGH)
void fanOn() {
    digitalWrite(fanPin, HIGH);  // Liga a primeira ventoinha
    digitalWrite(fanPinTwo, HIGH);  // Liga a segunda ventoinha
}

// Desliga a ventoinha (GPIO LOW)
void fanOff() {
    digitalWrite(fanPin, LOW);  // Desliga a primeira ventoinha
    digitalWrite(fanPinTwo, LOW);  // Desliga a segunda ventoinha
}

void loop() {
    Blynk.run();  // Mantém a conexão com a plataforma Blynk
    updateDisplay();
    if (fanPinState == LOW && fanPinTwoState == LOW) {
        fanController();  // Controla as ventoinhas com base na temperatura
```

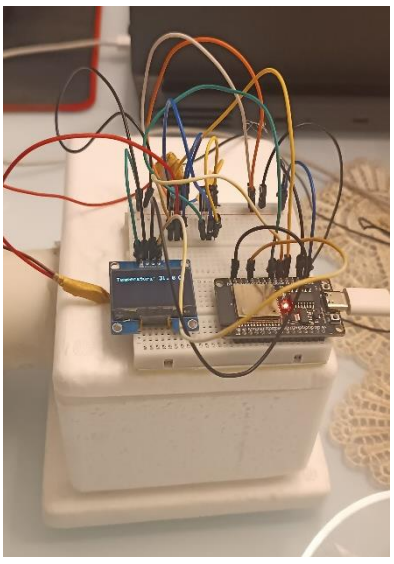
- **Testes e Validação**

A validação do sistema foi realizada por meio de testes em condições reais para avaliar o desempenho conforme os requisitos especificados. Foram realizados testes de integração, resposta do sistema às variações de temperatura e precisão do sensor DHT-11. A seguir, são apresentados os resultados obtidos.

Figura 3,4 e 5– Testes finais do projeto







2. RESULTADOS

2.1 Desempenho do Sistema

Os testes realizados demonstraram que o sistema de refrigeração automatizado funcionou conforme o esperado, mantendo a temperatura interna da caixa dentro dos parâmetros predefinidos. A integração com a plataforma Blynk permitiu controle remoto eficiente, possibilitando monitoramento e ajustes em tempo real. O sistema foi capaz de responder rapidamente às variações de temperatura, acionando as ventoinhas de forma eficaz.

2.2 Avaliação dos Componentes

Microcontroladora ESP32: Mostrou-se adequada para a aplicação, proporcionando conectividade e capacidade de processamento suficientes para a integração com os componentes e a plataforma Blynk.

Sensor de Temperatura DHT-11: Apresentou uma precisão de ±1°C em condições normais, adequada para a maioria das aplicações. No entanto, em condições de alta umidade, a precisão diminuiu para ±2°C.

Ventoinhas: Foram eficientes na manutenção da temperatura, com um tempo médio de resposta de 2 segundos após a detecção de uma temperatura acima do valor predefinido.

Display OLED: Proporcionou uma visualização clara e precisa da temperatura, facilitando o monitoramento em tempo real.

2.3 Consumo de Energia

O sistema consumiu em média 500 mA em operação contínua, com picos de 700 mA durante o acionamento das ventoinhas. Esse consumo é considerado eficiente para a aplicação proposta, mas há espaço para melhorias em futuras versões do projeto.

2.4 Discussão

Comparação com Sistemas Convencionais

O sistema desenvolvido apresenta várias vantagens em relação aos sistemas de controle de temperatura convencionais. A automação completa elimina a necessidade de intervenção manual constante, reduzindo a possibilidade de erros humanos. A capacidade de controle

remoto através da plataforma Blynk oferece maior flexibilidade e acessibilidade, características não comuns em sistemas tradicionais.

2.5 Desafios e Melhorias Futuras

Os principais desafios enfrentados durante o desenvolvimento incluem a necessidade de otimizar o consumo de energia das ventoinhas e a melhoria da precisão do sensor DHT-11 em condições extremas. Para versões futuras, recomenda-se a adoção de ventoinhas mais eficientes e a calibração do sensor para melhor desempenho em diferentes condições ambientais. Além disso, a implementação de algoritmos de controle mais avançados, como controle PID, poderia melhorar a precisão e a eficiência do sistema.

2.6 Impacto Ambiental e Sustentabilidade

O uso de um sistema automatizado de controle de temperatura contribui significativamente para a sustentabilidade ambiental. A automação reduz o consumo de eletricidade, diminuindo a pegada de carbono associada à operação do sistema. A capacidade de monitoramento remoto permite ajustes precisos, evitando o desperdício de energia.

3. Considerações Finais

O projeto desenvolvido resultou em um sistema de refrigeração automatizado funcional e eficiente, capaz de manter a temperatura de uma caixa de isopor dentro dos parâmetros desejados. A integração com a plataforma Blynk adicionou uma camada de flexibilidade e conveniência, permitindo o controle remoto do sistema. Este projeto não apenas demonstrou a viabilidade técnica da criação de um sistema de controle de temperatura eficaz para uma caixa de isopor, mas também destacou a integração de várias tecnologias modernas, incluindo sensores, atuadores e plataformas de controle remoto, como a Blynk.

Os resultados obtidos durante os testes confirmam que o sistema é capaz de manter a temperatura interna da caixa de isopor dentro dos parâmetros desejados de forma autônoma e eficiente. A capacidade de monitoramento e controle remoto via Blynk adiciona uma camada significativa de conveniência e flexibilidade, permitindo que usuários ajustem e monitorem a temperatura em tempo real a partir de qualquer lugar.

Além dos benefícios práticos, o projeto ofereceu uma oportunidade valiosa para os alunos aplicarem conhecimentos teóricos em um contexto real, enfrentando desafios e

desenvolvendo soluções inovadoras. A escolha do ESP32 se mostrou particularmente acertada devido à sua versatilidade e capacidades de conectividade, enquanto o sensor DHT11, apesar de suas limitações em condições extremas, provou ser adequado para a maioria das situações.

REFERÊNCIAS

THOMSEN, Adilson. Monitorando Temperatura e Umidade com o sensor DHT11. Maker Hero, 2013. Disponível em: https://www.makerhero.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/. Acesso em: 01 de junho de 2024.

SANTOS, Felipe. Aprenda a configurar a rede WiFi do ESP32 pelo smartphone. Maker Hero, 2020. Disponível em: https://www.makerhero.com/blog/aprenda-a-configurar-a-rede-wifi-do-esp32-pelo-smartphone/. Acesso em: 01 de junho de 2024.

BRINCANDO COM O ESP32 NO ARDUINO IDE. Tudo sobre IOT, 2022. Disponível em:https://www.tudosobreiot.com.br/blog/1098-iot-feito-facil_-brincando-com-o-esp32-no-arduino-ide. Acesso em: 01 de junho de 2024.

FREITAS, Gabriel. Blynk IoT – Monitoramento com ESP8266 e Sensor de Umidade do Solo (Higrômetro). Smart Kits, 2021. Disponível em: https://blog.smartkits.com.br/blynk-iot-monitoramento-com-esp8266-e-sensor-de-umidade-do-solo-higrometro/. Acesso em: 01 de junho de 2024.