**UNIVERSIDADE DE MARÍLIA (UNIMAR)**

**Bacharelado em Ciências da Computação**

Fábrica de Projetos – II

**Climatização Inteligente: Desenvolvendo Soluções Inovadoras em Ar Condicionado**

**Integrantes:** Alex Veloso, Breno Henrique, Caio Henrique, Cauã Santos,

Gustavo Soares, Pedro Bernardo, Leandro Poletti

**INTRODUÇÃO DO PROJETO**

A automação residencial está redefinindo a maneira como interagimos com nossos dispositivos em casa, proporcionando ambientes mais inteligentes e eficientes. Um dos aparelhos fundamentais para o conforto doméstico é o ar condicionado. No entanto, muitos sistemas de ar condicionado tradicionais apresentam restrições em termos de controles e flexibilidade, deixando os usuários em busca de soluções mais adaptáveis e personalizadas. É nesse contexto que este projeto ganha vida. Nossa proposta é criar um sistema de automação de ar condicionado baseado na plataforma Arduino, um micro controlador versátil e acessível. Este sistema é projetado para oferecer um controle avançado e altamente personalizado do sistema de climatização residencial, elevando o nível de conforto e eficiência. Além disso, ele traz consigo uma série de benefícios notáveis. Ao permitir que os usuários controlem o ar condicionado através de seus dispositivos móveis e computadores, oferecemos um novo patamar de comodidade. Isso não apenas simplifica a operação do aparelho, mas também permite que os usuários ajustem o ambiente a partir de qualquer lugar, fornecendo uma experiência verdadeiramente flexível. Além disso, este projeto abre portas para a criação de um ecossistema integrado de dispositivos domésticos inteligentes. O ar condicionado pode se comunicar e colaborar com outros dispositivos, Essa integração resulta em uma casa mais eficiente, conveniente e personalizada. Neste relatório, exploraremos detalhadamente o desenvolvimento e a implementação dessa solução, destacando como ela está redefinindo a climatização residencial e oferecendo uma série de benefícios significativos para os usuários e o ambiente doméstico.

**OBJETIVO DO PROJETO:**

O objetivo primordial deste projeto é aprimorar o controle e a eficiência do sistema de ar condicionado, por meio da integração do Arduino como plataforma central de controle. A abordagem consiste em reconfigurar o micro controlador Arduino para assumir as funções tradicionalmente atribuídas ao controle convencional do ar-condicionado, com um foco especial em aprimorar o controle remoto e automático das operações.

A transformação engloba a capacidade do Arduino de executar as tarefas fundamentais, tais como ligar e desligar o sistema de ar condicionado de forma remota, bem como ajustar com precisão a temperatura desejada e a velocidade do ventilador. Esse controle avançado e integrado permitirá que os usuários gerenciem o ambiente climatizado de maneira mais conveniente e eficiente, contribuindo para uma experiência aprimorada de conforto e economia de energia.

**JUSTIFICATIVA E PROPOSTAS CORRELATAS**

A climatização da sala de aula por meio de um aplicativo online oferece vantagens significativas, como a personalização do aprendizado, o monitoramento em tempo real da temperatura ambiente da sala de aula. A automação de sistemas de ar condicionado utilizando o Arduino como plataforma central de controle oferece uma série de benefícios e se alinha com diversas necessidades e tendências na área de climatização e automação residencial.

A iniciativa de automatizar sistemas de ar condicionado utilizando o Arduino como plataforma central de controle está alinhada com tendências significativas na área de automação residencial e Internet das Coisas (IoT). Essa abordagem inovadora pode ser contextualizada em projetos anteriores que exploram a automação de ambientes e oferecem insights valiosos sobre como a tecnologia pode melhorar a qualidade de vida e a eficiência energética.

A proposta de automatizar sistemas de ar condicionado do ambiente por meio de aplicativos online tem sido objeto de interesse em diversos projetos já existentes como: Smart Place 3.0: Gerenciamento Automático de Ar Condicionados em Ambientes Complexos, Sistema automatizado de gerenciamento de ar condicionado usa sensores e câmeras para ajustar a temperatura com base na presença de pessoas, economizando energia em ambientes complexos. Projeto Ubique: Sistema de monitoramento e controle de ar-condicionado. Este projeto visa reduzir o consumo de energia nos aparelhos de ar-condicionado do Centro de Informática, controlando-os remotamente com um sistema de baixo custo que utiliza sensores e uma placa micro controladora conectada à internet via WiFi e protocolo MQTT.

**ADERÊNCIA AS DISCIPLINAS**

Este projeto de automação de ar condicionado com Arduino demonstra uma forte aderência aos conceitos de Programação Orientada a Objetos, Estrutura de Dados e Sistemas Digitais, O projeto utiliza criação de classes e objetos para representar os componentes do sistema, como sensores, atuadores e controles de temperatura uso de estruturas de dados, como ponteiros, structs e estruturas lineares, para armazenar e manipular informações relevantes, como dados de sensores e configurações do sistema e o projeto incorpora conceitos de sistemas digitais ao utilizar o Arduino como plataforma central de controle

**PROTOTIPAÇÃO**

Materiais Utilizados

-Arduino Uno: Utilizamos o Arduino Uno como a unidade de controle central para o projeto.

-Módulo Sensor de Umidade e Temperatura DHT11: Esse sensor é utilizado para medir a temperatura e umidade ambiente.

-Protoboard e Resistores: Usados para montar os circuitos de forma organizada e para fornecer resistência elétrica quando necessário.

-Kit de Jumpers: Os jumpers são usados para conectar os componentes de forma elétrica.

-LED Emissor de Infravermelho: O LED é usado para enviar comandos infravermelhos para o ar condicionado.

-Sensor IRM e Receptor Infravermelho: O sensor IRM é usado para capturar comandos infravermelhos do controle remoto do ar condicionado.

-Módulo Transmissor de Infravermelho: Usado para enviar os comandos infravermelhos para o ar condicionado.

-Módulo Emissor de Infravermelho: Usado para emitir sinais infravermelhos que são recebidos pelo ar condicionado.

-Módulo de Interface de Display LCD para Arduino: Utilizamos um display LCD para exibir informações como temperatura e status do ar condicionado.

-Placa ESP32: Utilizada para habilitar a conectividade Wi-Fi e a integração com a IoT

**FUNCIONALIDADES:**

-Medição de Temperatura e Umidade: O sensor DHT11 mede a temperatura e umidade ambiente e exibe os valores no display LCD.

-Controle Remoto do Ar Condicionado: O sistema é capaz de capturar comandos infravermelhos do controle remoto original do ar condicionado usando o sensor IRM. Ele também envia comandos infravermelhos para o ar condicionado por meio do módulo transmissor de infravermelho.

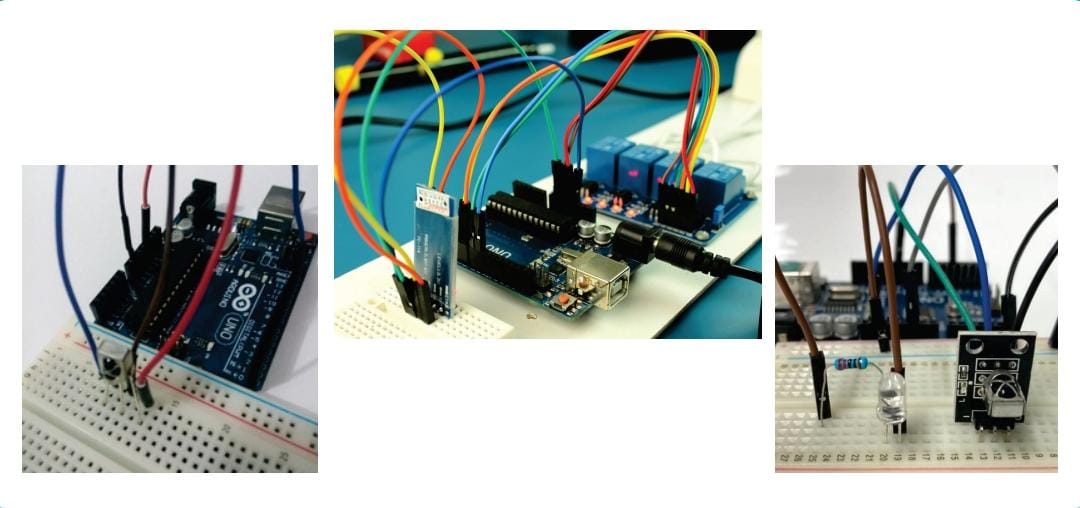
-Controle por Aplicativo IoT: Um aplicativo móvel permite que os usuários controlem o ar condicionado remotamente. Os comandos são enviados para a placa ESP32, que os encaminha para o Arduino.

-Exibição no Display: O display LCD mostra a temperatura atual, umidade e o status do ar condicionado (ligado/desligado)

**CRONOGRAMA DE EXECUÇÃO**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Semanas** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **Idealização do Projeto** | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Pesquisa de projetos relacionados** |  |  | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Busca por Materiais** |  |  | **X** | **X** |  |  |  |  |  |  |  |  |  |  |  |  |
| **Elaboração do código** |  |  |  |  | **X** | **X** | **X** |  |  |  |  |  |  |  |  |  |
| **Testes no TinkerCad** |  |  |  |  | **X** | **X** | **X** | **X** |  |  |  |  |  |  |  |  |
| **Apresentação** |  |  |  |  |  |  | **X** | **X** |  |  |  |  |  |  |  |  |
| **Montagem Do Arduino** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Desenvolvimento do App** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Teste do App** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Revisão final do projeto** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Apresentação Final** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**EXPECTATIVA DO PROJETO:**

****

****

**APENDICE**

**CODIGO EM C++ NO APLICATIVO TINKERCAD**

#include <LiquidCrystal.h>

class DisplayLCD {

private:

LiquidCrystal lcd;

.

public:

DisplayLCD(int rs, int en, int d4, int d5, int d6, int d7)

: lcd(rs, en, d4, d5, d6, d7) {}

void begin(int cols, int rows) {

lcd.begin(cols, rows);

}

void clear() {

lcd.clear();

}

void print(String message) {

lcd.print(message);

}

void setCursor(int col, int row) {

lcd.setCursor(col, row);

}

};

class ControladorTemperatura {

private:

LiquidCrystal lcd;

const int pinoTemperatura;

const int pinoBotaoMais;

const int pinoBotaoMenos;

const int pinoBotaoModo;

const int pinoLED;

const int pinoMotor;

float temperaturaInstantanea;

float temperaturaCalculada;

float temperaturaAlvo;

float faixaTemperatura;

float leiturasTemperatura[10];

int intervaloAtraso;

int modoTela;

int indiceAtual;

unsigned long ultimoTempoLeitura;

public:

ControladorTemperatura(int tempPin, int btnMaisPin, int btnMenosPin, int btnModoPin, int lPin, int mPin)

: lcd(12, 11, 5, 4, 3, 2),

pinoTemperatura(tempPin),

pinoBotaoMais(btnMaisPin),

pinoBotaoMenos(btnMenosPin),

pinoBotaoModo(btnModoPin),

pinoLED(lPin),

pinoMotor(mPin),

temperaturaAlvo(25.0),

faixaTemperatura(2.0),

intervaloAtraso(350),

modoTela(0),

indiceAtual(0),

ultimoTempoLeitura(0) {}

void configuracao() {

Serial.begin(9600);

lcd.begin(16, 2);

lcd.clear();

pinMode(pinoBotaoMais, INPUT\_PULLUP);

pinMode(pinoBotaoMenos, INPUT\_PULLUP);

pinMode(pinoBotaoModo, INPUT\_PULLUP);

pinMode(pinoLED, OUTPUT);

pinMode(pinoMotor, OUTPUT);

digitalWrite(pinoMotor, HIGH);

digitalWrite(pinoLED, LOW);

for (int i = 0; i < 10; i++) {

leiturasTemperatura[i] = map(analogRead(pinoTemperatura), 20, 358, -40, 125);

}

temperaturaCalculada = calcularTemperaturaMedia();

}

void ciclo() {

lerSensoresTemperatura();

switch (modoTela) {

case 0:

exibirTemperaturaAtual();

exibirTemperaturaAlvo();

ajustarTemperatura();

break;

case 1:

exibirTelaConfiguracao();

ajustarFaixaTemperatura();

break;

case 2:

exibirTemperaturaInstantanea();

break;

default:

break;

}

if (digitalRead(pinoBotaoModo) == LOW) {

atrasoClique();

modoTela = 1;

}

delay(300);

}

private:

float calcularTemperaturaMedia() {

float soma = 0;

for (int i = 0; i < 10; i++) {

soma += leiturasTemperatura[i];

}

return soma / 10;

}

void lerSensoresTemperatura() {

if (millis() - ultimoTempoLeitura > 10000) {

for (int i = 0; i < 9; i++) {

leiturasTemperatura[i] = leiturasTemperatura[i + 1];

}

leiturasTemperatura[9] = map(analogRead(pinoTemperatura), 20, 358, -40, 125);

temperaturaCalculada = calcularTemperaturaMedia();

ultimoTempoLeitura = millis();

}

}

void exibirTemperaturaAtual() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp Atual: ");

lcd.setCursor(12, 0);

lcd.print(temperaturaCalculada);

}

void exibirTemperaturaAlvo() {

lcd.setCursor(0, 1);

lcd.print("Def Temp: ");

lcd.setCursor(12, 1);

lcd.print(temperaturaAlvo);

}

void exibirTelaConfiguracao() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Tela Config.");

lcd.setCursor(0, 1);

lcd.print("Limite: ");

lcd.setCursor(12, 1);

lcd.print(faixaTemperatura);

}

void exibirTemperaturaInstantanea() {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Temp. Instant.");

lcd.setCursor(0, 1);

lcd.print(temperaturaInstantanea);

}

void ajustarTemperatura() {

if (digitalRead(pinoBotaoMais) == LOW) {

temperaturaAlvo += 0.5;

atrasoClique();

}

if (digitalRead(pinoBotaoMenos) == LOW) {

temperaturaAlvo -= 0.5;

atrasoClique();

}

}

void ajustarFaixaTemperatura() {

if (digitalRead(pinoBotaoMais) == LOW) {

faixaTemperatura += 0.5;

atrasoClique();

}

if (digitalRead(pinoBotaoMenos) == LOW) {

faixaTemperatura -= 0.5;

atrasoClique();

}

}

void atrasoClique() {

delay(100);

}

};

ControladorTemperatura controladorTemperatura(A0, 7, 8, 10, 9, 6);

void setup() {

controladorTemperatura.configuracao();

}

void loop() {

controladorTemperatura.ciclo();

}

**Diagrama**

@startuml

class temperatura {

 -unit: celsius

  float: valor

}

class displayLCD {

 +escrever(string)

 -cor: azul

}

note right

informa a tempertaura ambiente

end note

temperatura <.. sensor

sensor <|-- displayLCD

note right

coleta dados ambiente

end note

@enduml