[GIMS]_ArmarioInteligente_TDE01

Trabalho Docente Estruturante - TDE 01

Projeto: Armário Inteligente utilizando ESP32

Integrantes:

Gabriel Carmo

Igor Thiago

Marcel Neto

Samuel Pereira

1. Objetivo do Projeto

Desenvolver uma solução de controle de acesso para armários de notebooks da PUC, utilizando ESP32 como unidade central de processamento. O sistema visa:

- Garantir o acesso apenas a alunos e professores autorizados através de autenticação biométrica.

- Monitorar a presença dos notebooks dentro das gavetas via sensores de presença.

- Alertar em tempo real, através de LEDs e notificação à plataforma online, sobre usos não autorizados.

- Desenvolver uma plataforma web para monitoramento remoto dos acessos e registros de eventos.

2. Justificativa

A segurança e a organização dos recursos tecnológicos são essenciais no ambiente acadêmico. O Armário Inteligente propõe uma solução ciberfísica moderna, baseada em autenticação biométrica e IoT, reduzindo riscos de furto, uso indevido e possibilitando gestão remota eficiente dos equipamentos.

3. Tecnologias Utilizadas

Hardware:

- ESP32 DevKit v1

- Sensor Infravermelho (detecção de presença)

- Leitor Biométrico R307

- LEDs (Verde, Vermelho e Amarelo)

- Protoboard, jumpers e cabos

Software:

- Arduino IDE (linguagem C/C++)

- Node.js (Servidor HTTP)

- HTML/CSS (Interface Web)

Protocolos de Comunicação:

- Wi-Fi IEEE 802.11 b/g/n

- HTTP (POST)

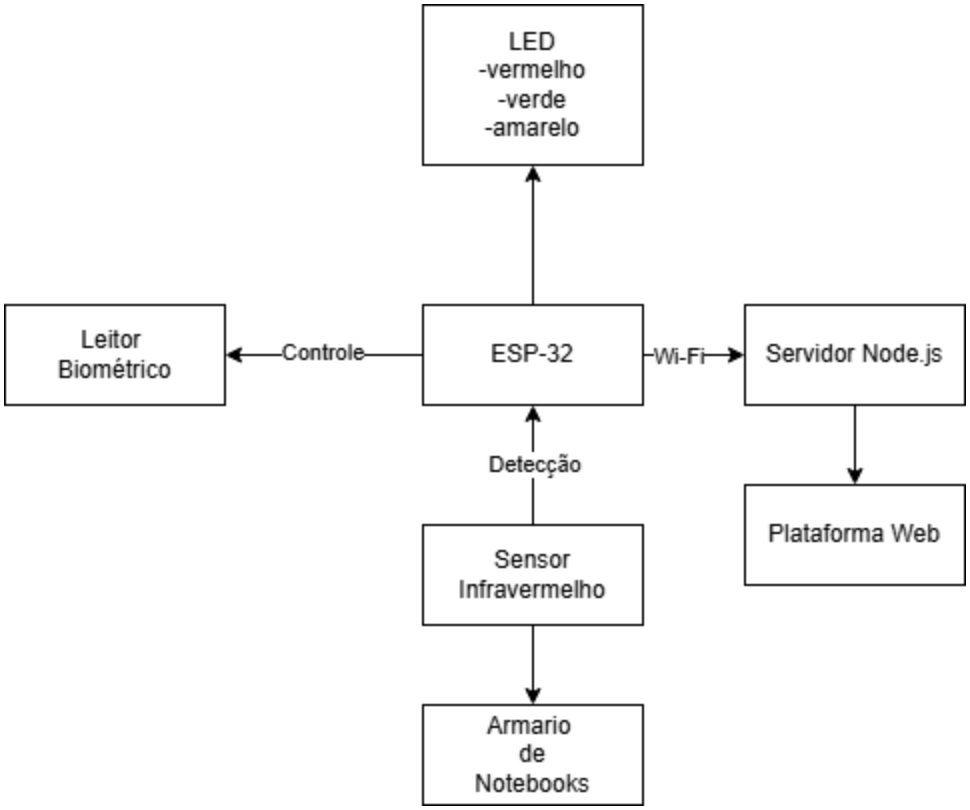
4. Arquitetura Geral do Sistema

Usuário → Leitor Biométrico → ESP32

Sensor IR → ESP32

ESP32 → LEDs de Status

ESP32 → (Wi-Fi) → Servidor Node.js → Plataforma Web



5. Cronograma de Execução

Semana 1: Definição do projeto e estudo de componentes

Semana 2: Teste do leitor biométrico e sensor infravermelho

Semana 3: Teste dos Leds e comunicação wifi para enviar dados ao servidor

Semana 4: Desenvolvimento do servidor Node.js

Semana 5: Desenvolvimento da página web

Semana 6: Integração dos módulos

Semana 7: Testes finais e ajustes

Semana 8: Entrega da TDE 01

6. Testes Isolados de Sensores, Atuadores e Módulos

6.1 Leitor Biométrico

Objetivo: Verificar a captura e reconhecimento de digitais.

6.2 Sensor Infravermelho

Objetivo: Detectar presença ou ausência do notebook.

6.3 LEDs

Objetivo: Verificar a funcionalidade dos LEDs.

6.4 Wi-Fi e Servidor Node.js

Objetivo: Testar envio de dados via HTTP.

7. Uso do Git e Organização da Equipe

Link do github: https://github.com/IgorThiagoLara/PSC-Armario_inteligente.git

- Repositório GitHub estruturado.

- Commits frequentes e comentados.

8. Documentação Inicial (Anexos)

- Códigos de testes isolados

- Diagrama da arquitetura

Códigos de Testes Isolados

Teste do Leitor Biométrico

```
#include <Adafruit_Fingerprint.h>
#include <HardwareSerial.h>

HardwareSerial mySerial(2);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup() {
  Serial.begin(115200);
  mySerial.begin(57600);
  if (finger.begin()) {
    Serial.println("Leitor biométrico iniciado!");
  } else {
    Serial.println("Erro no leitor biométrico!");
    while (1) { delay(1); }
  }
}

void loop() {
  getFingerprintID();
  delay(100);
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK) return p;
  p = finger.image2Tz();
  if (p != FINGERPRINT_OK) return p;
  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK) {
    Serial.print("ID encontrado: ");
    Serial.println(finger.fingerID);
  } else {
    Serial.println("Digital não reconhecida.");
  }
  return p;
}
```

Teste do Sensor Infravermelho

```
#define SENSOR_IR_PIN 34

void setup() {
  Serial.begin(115200);
  pinMode(SENSOR_IR_PIN, INPUT);
}
```

```
void loop() {
  int sensorValue = digitalRead(SENSOR_IR_PIN);
  if (sensorValue == LOW) {
    Serial.println("Notebook presente.");
  } else {
    Serial.println("Notebook ausente!");
  }
  delay(500);
}
```

Teste dos LEDs

```
#define LED_VERDE 26
#define LED_VERMELHO 27
#define LED_AMARELO 25

void setup() {
  Serial.begin(115200);
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);
  pinMode(LED_AMARELO, OUTPUT);
}

void loop() {
  digitalWrite(LED_VERDE, HIGH);
  delay(1000);
  digitalWrite(LED_VERDE, LOW);
  digitalWrite(LED_VERMELHO, HIGH);
  delay(1000);
  digitalWrite(LED_VERMELHO, LOW);
  digitalWrite(LED_AMARELO, HIGH);
  delay(1000);
  digitalWrite(LED_AMARELO, LOW);
}
```

Teste de Comunicação Wi-Fi

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "visitantes";
const char* password = "";
const char* serverName = "http://armario_inteligente:8000/dados";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("Wi-Fi conectado!");
}

void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");
    String jsonData = "{"teste":"Mensagem de Teste do ESP32"}";
    int httpResponseCode = http.POST(jsonData);
    Serial.println(httpResponseCode);
    http.end();
  }
  delay(5000);
}
```