

IFCE - Instituto Federal do Ceará

Campus Maracanaú

Relatório - Projeto prático 01: Sistema de casa inteligente.

Disciplina: Microcontroladores

Professor: Pedro Hericson Machado

Equipe: Francisco Aldenor, Elías de Almeida e Isabelli Pinheiro

Introdução

Nesta atividade prática realizada com microcontroladores, o objetivo foi desenvolver um protótipo de casa inteligente utilizando o ESP32. O projeto contemplou duas funcionalidades principais: um sistema de controle de acesso via cartão magnético e um monitor de nível da caixa d'água com alerta visual. Ambas as funcionalidades foram integradas com um bot do Telegram para gerenciamento remoto e log dos eventos em memória persistente.

Metodologia

Componentes do Circuito

- 1 ESP32 (30 pinos)
- 1 cabo micro USB
- 1 protoboard
- 1 sensor de distância HC-SR04
- 1 leitor RFID RDM6300
- 1 relé
- 4 LEDs (verde, amarelo, vermelho e azul)
- 13 cabos jumper macho-fêmea
- 4 cabos jumper macho-macho
- 3 cabos jumper fêmea-fêmea

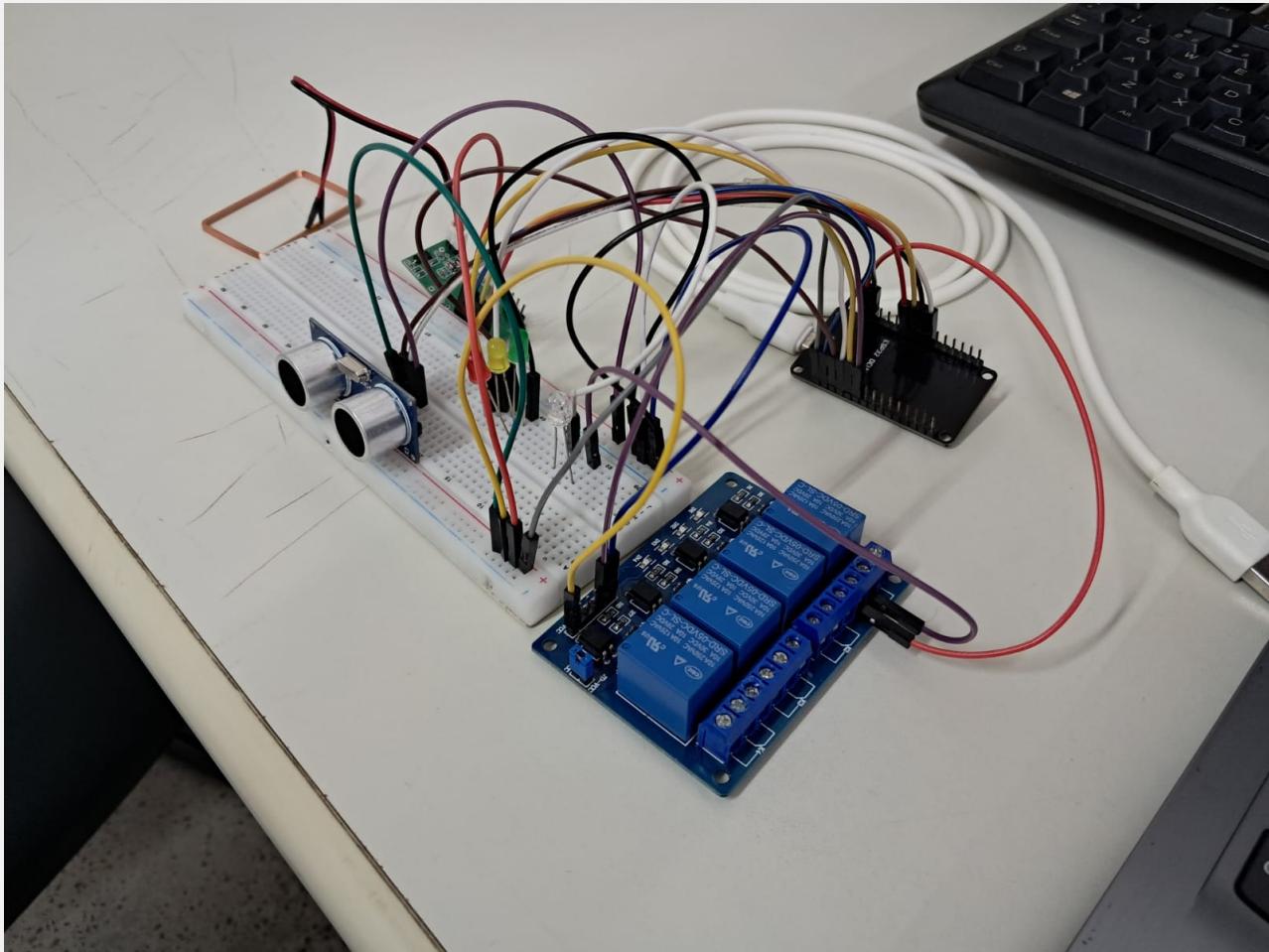
Conexões

Abaixo estão as conexões principais:

- O sensor RFID foi conectado ao ESP32 via portas GPIO16 (TXD2) e GPIO17 (RXD2).
- O relé foi acionado pelo GPIO27 e ligado ao LED azul, simulando uma fechadura.
- O 5v do RFID foi conectado ao VIN do ESP e o GND do RFID ao GND do ESP.
- Os LEDs verde, amarelo e vermelho foram conectados aos GPIOs 13, 12 e 14, respectivamente, para indicar o nível de água.
- O sensor de distância HC-SR04 foi conectado aos GPIOs 5 (trigger) e 18 (echo).

- O GCC do sensor de distância HC-SR04 foi conectado ao VIN do ESP e o GND do sensor de distância HC-SR04 ao GND do ESP.

📷 Imagem do circuito montado:

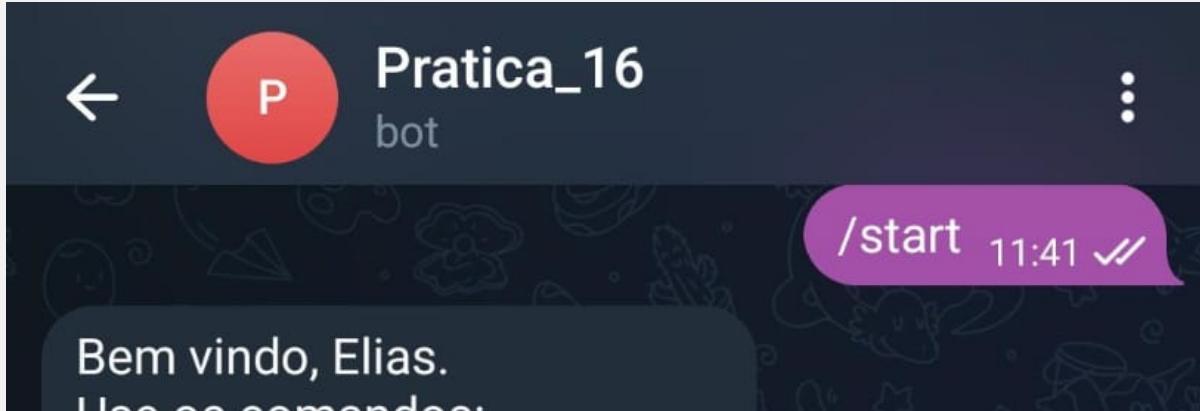


Funcionamento

1. Sistema de Acesso com RFID

- Leitura de cartões magnéticos via sensor RFID.
- Verificação do cartão: se autorizado, aciona o relé e acende o LED azul(simulando a abertura da porta).
- Se o cartão for não autorizado, exibe mensagem de acesso negado no terminal.
- Cadastro e remoção de cartões são feitos remotamente através de comandos no bot do Telegram.

📷 Comandos via Telegram:



use os comandos.
/cadastrar <codigo>
/descadastrar <codigo>
/definir_nivel_verde <cm>
/definir_nivel_amarelo <cm>
/definir_nivel_vermelho <cm>
/log

11:41

/cadastrar 5100D8695F

11:44 ✓

Cartão cadastrado: 5100D8695F

11:44

/descadastrar 5100D8B265

11:45 ✓

Cartão removido: 5100D8B265

11:45

/log 11:47 ✓

#3462 [06/06/2025 11:47:02] Medição:
70.62 cm#3463 [06/06/2025 11:47:03] Medição:
70.62 cm#3464 [06/06/2025 11:47:05] Medição:
70.62 cm#3465 [06/06/2025 11:47:07] Medição:
70.60 cm#3466 [06/06/2025 11:47:08] Medição:
70.62 cm#3467 [06/06/2025 11:47:10] Medição:
70.60 cm

#3468 [06/06/2025 11:47:12] Medição:



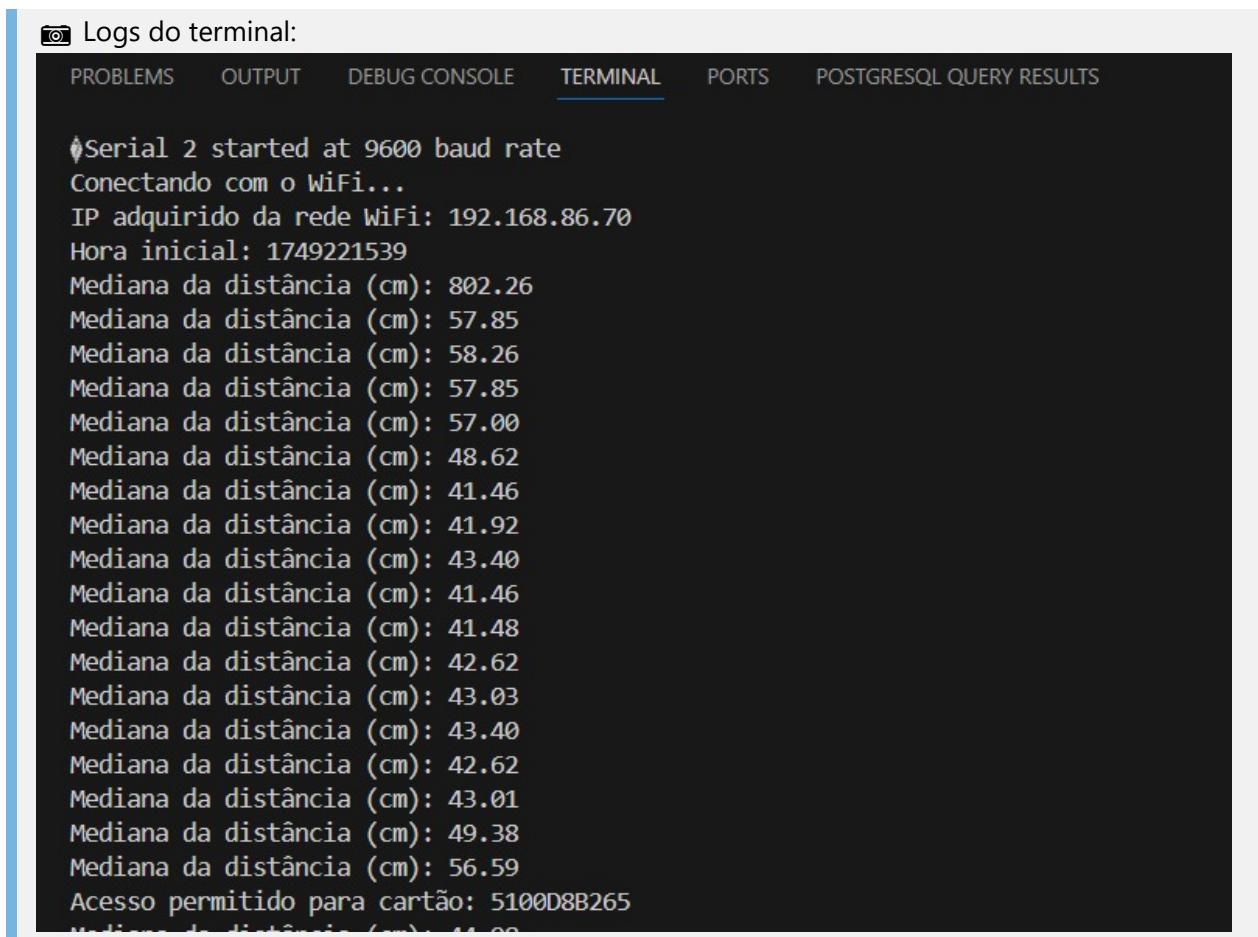
Mensagem



- Utilização de sensor ultrassônico HC-SR04 para medir a distância da água.
- Níveis indicados visualmente com LEDs:
 - Verde: nível cheio (até 10cm)
 - Amarelo: intermediário (10cm a 20cm)
 - Vermelho: baixo (20cm a 30cm)
 - Sem LED: Acima de 30 cm(leitura inválida por distância excedida)

3. Armazenamento de Logs

- Todos os eventos de leitura de cartão e medições da caixa d'água são salvos em um arquivo no sistema de arquivos LittleFS do ESP32.



```
Serial 2 started at 9600 baud rate
Conectando com o WiFi...
IP adquirido da rede WiFi: 192.168.86.70
Hora inicial: 1749221539
Mediana da distância (cm): 802.26
Mediana da distância (cm): 57.85
Mediana da distância (cm): 58.26
Mediana da distância (cm): 57.85
Mediana da distância (cm): 57.00
Mediana da distância (cm): 48.62
Mediana da distância (cm): 41.46
Mediana da distância (cm): 41.92
Mediana da distância (cm): 43.40
Mediana da distância (cm): 41.46
Mediana da distância (cm): 41.48
Mediana da distância (cm): 42.62
Mediana da distância (cm): 43.03
Mediana da distância (cm): 43.40
Mediana da distância (cm): 42.62
Mediana da distância (cm): 43.01
Mediana da distância (cm): 49.38
Mediana da distância (cm): 56.59
Acesso permitido para cartão: 5100D8B265
Mediana da distância (cm): 41.88
```

Código Principal

A prática foi implementada no arquivo `main.cpp`. O código completo inclui o uso de multitarefas (FreeRTOS), leitura de sensores, controle de GPIOs, conexão Wi-Fi, integração com o Telegram e manipulação de arquivos no sistema de arquivos interno.

Segue o código do arquivo `main.cpp`:

```
// Bibliotecas externas
#include <Arduino.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <FS.h>
#include <LittleFS.h>
```

```
#include <time.h>

// Cabeçalhos dos arquivos do projeto
#include "config.h"
#include "distance_sensor.h"
#include "log_util.h"
#include "rfid_card.h"
#include "telegram_bot.h"

// Caixa d'água
#define WATER_TANK_GREEN 13
#define WATER_TANK_YELLOW 12
#define WATER_TANK_RED 14
#define TRIG_PIN 5
#define ECHO_PIN 18
#define NUM_MEASUREMENTS 51

float greenLevel = 10;
float yellowLevel = 20;
float redLevel = 30;
float distances[NUM_MEASUREMENTS];

// Fechadura da porta
#define DOOR_LOCK 27
#define RXD2 16
#define TXD2 17
#define BAUD 9600

HardwareSerial RFID(2);
String cardSerial;
time_t lastTimeReading = 0;

// Bot do Telegram
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);

unsigned long lastTimeBotRan;
int botRequestDelay = 1000;

// Protótipos das tasks
void waterTankMeasuringTask(void *parameter);
void doorAccessTask(void *parameter);

void setup() {
    Serial.begin(9600);

    // Pinos do circuito
    pinMode(WATER_TANK_GREEN, OUTPUT);
    pinMode(WATER_TANK_YELLOW, OUTPUT);
    pinMode(WATER_TANK_RED, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(DOOR_LOCK, OUTPUT);
```

```
// Leitor RFID RDM6300
RFID.begin(BAUD, SERIAL_8N1, RXD2, TXD2);
Serial.println("Serial 2 iniciado com taxa de baud 9600");

// LittleFS
if (!LittleFS.begin()) {
    Serial.println("Erro ao montar LittleFS");
    return;
}

File logFile = LittleFS.open("/log.txt", "w");
if (logFile) {
    logFile.println("### LOG INICIALIZADO ###");
    logFile.close();
}

File indexFile = LittleFS.open("/log_index.txt", "r");
logIndex = indexFile ? indexFile.readStringUntil('\n').toInt() + 1 : 1;
indexFile.close();

if (!LittleFS.exists("/authorized_cards.txt")) {
    File cardsFile = LittleFS.open("/authorized_cards.txt", "w");
    if (cardsFile) {
        cardsFile.println("# Cartões cadastrados");
        cardsFile.close();
    }
}

// Rede WiFi
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);

#ifndef ESP32
    client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
#endif

while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando com o WiFi...");
}

Serial.print("IP adquirido da rede WiFi: ");
Serial.println(WiFi.localIP());

// Horário
configTime(-3 * 3600, 0, "pool.ntp.org", "time.nist.gov");
struct tm timeinfo;
while (!getLocalTime(&timeinfo)) {
    Serial.println("Aguardando sincronização NTP...");
    delay(1000);
}

time(&lastTimeReading);
Serial.print("Hora inicial: ");
```

```
Serial.println(lastTimeReading);

// Tasks
xTaskCreatePinnedToCore(waterTankMeasuringTask, "WaterTank", 4096, NULL, 1,
NULL, 0);
xTaskCreatePinnedToCore(doorAccessTask, "DoorAcess", 4096, NULL, 1, NULL, 1);
}

void loop() {
// Recebe comandos do bot do Telegram
if (millis() > lastTimeBotRan + botRequestDelay) {
int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
while (numNewMessages) {
handleNewMessages(numNewMessages);
numNewMessages = bot.getUpdates(bot.last_message_received + 1);
}
lastTimeBotRan = millis();
}
}

// Task de medição da caixa d'água
void waterTankMeasuringTask(void *parameter) {
while (true) {
for (int i = 0; i < NUM_MEASUREMENTS; i++) {
distances[i] = measureDistance();
delay(5);
}

sortArray(distances, NUM_MEASUREMENTS);
float median = distances[NUM_MEASUREMENTS / 2];

logToFile("Medição: " + String(median) + " cm");
Serial.println("Mediana da distância (cm): " + String(median));

digitalWrite(WATER_TANK_GREEN, LOW);
digitalWrite(WATER_TANK_YELLOW, LOW);
digitalWrite(WATER_TANK_RED, LOW);

if (median < greenLevel) digitalWrite(WATER_TANK_GREEN, HIGH);
else if (median < yellowLevel) digitalWrite(WATER_TANK_YELLOW, HIGH);
else if (median < redLevel) digitalWrite(WATER_TANK_RED, HIGH);

delay(1000);
}
}
}

// Task de permissão da fechadura da porta
void doorAccessTask(void *parameter) {
while (true) {
while (RFID.available() > 0) {
char cardChar = RFID.read();
cardSerial += cardChar;
}
}
```

```
if (cardSerial.length() >= 14) {  
    String cleanSerial = cleanCardSerial(cardSerial);  
    String message;  
  
    if (isCardAuthorized(cleanSerial)) {  
        time_t currentTime;  
        time(&currentTime);  
  
        if (difftime(currentTime, lastTimeReading) >= 5) {  
            lastTimeReading = currentTime;  
            digitalWrite(DOOR_LOCK, LOW);  
            delay(2000);  
            digitalWrite(DOOR_LOCK, HIGH);  
            message = "Acesso permitido para cartão: " + cleanSerial;  
            Serial.println(message);  
            logToFile(message);  
        }  
    }  
    else {  
        message = "Acesso negado para cartão: " + cleanSerial;  
        Serial.println(message);  
        logToFile(message);  
    }  
    cardSerial = "";  
}  
  
delay(200);  
}  
}
```

Resultados

A prática foi bem-sucedida, com as seguintes observações:

- O sistema de controle de acesso reconheceu cartões cadastrados corretamente e negou os não autorizados.
- O relé funcionou perfeitamente simulando o acionamento de uma fechadura com o LED azul.
- O sensor de distância fez medições consistentes, e os LEDs representaram corretamente o nível da caixa d'água.
- Todos os eventos foram corretamente registrados em arquivos `log.txt` e `log_index.txt`.
- O sistema foi operado remotamente com sucesso por meio do bot no Telegram.

 **Demonstração em vídeo da prática:** [Assista no YouTube](#)

Conclusão

Esta prática permitiu a aplicação integrada de diversos conceitos de microcontroladores e IoT, como controle d e atuadores, leitura de sensores, comunicação remota via Telegram e registro de eventos. A arquitetura modular e a utilização de multitarefas com o ESP32 proporcionaram um funcionamento eficiente.