#define pinLED D4

int tentativas;

WiFi.mode(WIFI\_STA); //

WiFi.mode(WIFI\_AP);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED and tentativas <50)

{

tentativas++;

delay(250);

Serial.print(".");

digitalWrite(D4, HIGH);

delay(250);

digitalWrite(D4, LOW);

}

if (tentativas <50) {

IPAddress Ip(192,168,6,34);

IPAddress Gateway(192,168,6,1);

IPAddress Subnet(255,255,255,0);

WiFi.config(Ip, Gateway, Subnet);

Serial.println("");

Serial.println("WiFi conectado");

Serial.println(WiFi.localIP());

digitalWrite(5, HIGH);

}

else {

Serial.println("");

Serial.println(“Error de conexion”);

digitalWrite(15, LOW);

}

}

void loop()

{

Udp.beginPacket(“192.168.6.24”, 1234);

for(int i=0; i<1024;i++){

int old=micros();

newtime

oldtime

Definição dos pinos

#define LED\_PIN 2

#define LAMP\_PIN 0

#define BT\_PIN 3

// Configuração de IP da rede

IPAddress myIP(192,168,0,118);

IPAddress myDNS(192,168,0,1);

IPAddress myMASC(255,255,255,0);

void hardwareInit(void);

void trataRequest(void);

void enviaResposta(WiFiClient client, bool lampada, bool led);

void atualizaSaidas(bool lampada, bool led);

bool trataChave(void);

// Dados da rede WiFi

const char\* ssid = "######";

const char\* password = "\*\*\*\*\*\*";

const char\* headerHTLM = "HTTP/1.1 200 OK\nContent-Type: text/html\n\n<!DOCTYPE HTML>\n<html>\n";

// Corpo da resposta segue a estrutura -> [{Lampada:XX,Led:ZZ}]

const char\* tailHTML = "</html>\n";

bool statusLamp = 0;

bool funcaoLed = 1;

WiFiServer server(80);

void setup(){

Serial.begin(115200);

hardwareInit();

Serial.print("\nConectando a rede: ");

Serial.println(ssid);

WiFi.begin(ssid, password); // Conecta na rede

WiFi.config(myIP, myDNS, myMASC); // Configura IP estático

int i = 0;

while (WiFi.status() != WL\_CONNECTED){

delay(1000);

Serial.print(".");

if(++i > 30) ESP.restart(); // Reinicia ESP após 30 segundos

}

Serial.println("\nWiFi Conectado!");

server.begin(); // Inicia Server

Serial.print("Use essa URL para conectar: http://");

Serial.print(WiFi.localIP());

Serial.println("/");

}

void loop(){

trataRequest(); // Trata requisição do app

atualizaSaidas(statusLamp, funcaoLed);

if(trataChave()) statusLamp = !statusLamp;

}

void hardwareInit(void){ // Inicializa hardware

pinMode(LED\_PIN, OUTPUT);

pinMode(LAMP\_PIN, OUTPUT);

pinMode(BT\_PIN, INPUT);

digitalWrite(LED\_PIN, HIGH);

digitalWrite(LAMP\_PIN, HIGH);

}

void trataRequest(void){ // Trata requisição do app

WiFiClient client = server.available(); // Testa se foi feita requisição

if(!client) return;

while(!client.available()) delay(1); // Aguarda recepção dos dados

String request = client.readStringUntil('\r');// Le dados

client.flush();

if(request.indexOf("status") != -1){

enviaResposta(client, statusLamp, funcaoLed);// Responde requisição

return;

}

if(request.indexOf("lamp") != -1){

if(request.indexOf("on") != -1) statusLamp = true; // Liga lampada

if(request.indexOf("off") != -1) statusLamp = false; // Desliga lampada

enviaResposta(client, statusLamp, funcaoLed);

return;

}

if(request.indexOf("led") != -1){

if(request.indexOf("on") != -1) funcaoLed = true; // Liga função led

if(request.indexOf("off") != -1) funcaoLed = false; // Desliga função led

enviaResposta(client, statusLamp, funcaoLed);

return;

}

}

void enviaResposta(WiFiClient client, bool lampada, bool led){

// Monta corpo da resposta seguindo a estrutura -> [{Lampada:XX,Led:ZZ}]

String resposta = "[{Lampada:";

if(lampada) resposta = resposta + "ON";

else resposta = resposta + "OFF";

resposta += ",Led:";

if(led) resposta = resposta + "ON";

else resposta = resposta + "OFF";

resposta += "]}";

client.print(headerHTLM);

client.print(resposta);

client.print(tailHTML);

}

void atualizaSaidas(bool lampada, bool led){

if(lampada){

digitalWrite(LAMP\_PIN, LOW);

digitalWrite(LED\_PIN, HIGH);

} else{

digitalWrite(LAMP\_PIN, HIGH);

if(led) digitalWrite(LED\_PIN, LOW);

else digitalWrite(LED\_PIN, HIGH);

}

}

bool trataChave(void){

const unsigned long Delay = 50; // Variaveis para debouce da chave

static unsigned long tempoDebounce = 0;

static int estadoChave = HIGH, ultimaLeitura = HIGH;

bool flag = false;

int leitura = digitalRead(BT\_PIN);

if(leitura != ultimaLeitura) tempoDebounce = millis();

if((millis() - tempoDebounce) > Delay){

if(leitura != estadoChave){

estadoChave = leitura;

if(estadoChave == HIGH) flag = true;

}

}

ultimaLeitura = leitura;

return(flag);

}

void connect\_wifi(void){

//Inicia o WiFi

WiFi.begin(ssid, password);

Serial.print("Conectando");

//Loop até conectar no WiFi

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

//Logs na porta serial

Serial.println("");

Serial.println("WiFi conectado!");

Serial.print("Conectado na rede ");

Serial.println(WiFi.SSID());

Serial.print("IP: ");

Serial.println(WiFi.localIP());

Serial.print("Subnet Mask: ");

Serial.println(WiFi.subnetMask());

Serial.print("MAC Address: ");

Serial.println(WiFi.macAddress());

Serial.print("RSSI: ");

Serial.println(WiFi.RSSI());

digitalWrite(D6, HIGH);

delay(2000);

}

void setup(void){

//Configura o pino digital para saida/output

pinMode(D6, OUTPUT);

digitalWrite(D6, LOW);

//Configuração da UART

Serial.begin(115200);

//Logs na porta serial

Serial.print("Conectando na rede ");

Serial.println(ssid);

connect\_wifi();

Serial.println("Desabilitando o WiFi - Modo WIFI\_OFF");

WiFi.mode(WIFI\_OFF);

Serial.println("Forcando o desligamento do WiFi");

WiFi.forceSleepBegin();

delay(10);

digitalWrite(D6, LOW);

Serial.println("Forcando o religamento do WiFi");

WiFi.forceSleepWake();

Serial.println("Conectando na rede");

connect\_wifi();

}

void loop(void){

}