Utilizando o protocolo ESP-NOW do ESP32 para estabelecer conexão e acender LED de outro circuito

Elias de Almeida Sombra Neto[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) - Campus Maracanaú
Av. Parque Central, 1315 - Distrito Industrial I, Maracanaú-CE, Brasil

elias.almeida09@aluno.ifce.edu.br

Abstract. *This paper presents the implementation of a communication system between two ESP32 boards using the ESP-NOW protocol. The project was developed at IFCE during the Microcontrollers course, aiming to remotely control an LED through message exchange without Wi-Fi. The circuit was built with basic components, and the code was written in C++ using PlatformIO. Tests confirmed the protocol's efficiency in enabling direct communication between devices, successfully meeting the project goal.*

Resumo. *Este trabalho descreve a implementação de um sistema de comunicação entre dois ESP32 usando o protocolo ESP-NOW. A atividade foi realizada no IFCE durante a disciplina de Microcontroladores, com o objetivo de acionar um LED remotamente por meio da troca de mensagens sem utilizar Wi-Fi. O circuito foi montado com componentes básicos, e o código foi desenvolvido em C++ com PlatformIO. Os testes demonstraram a eficácia do protocolo na comunicação direta entre os dispositivos, atingindo com sucesso o objetivo proposto.*

1. Introdução

Este relatório descreve uma atividade prática realizada no Laboratório de Eletroeletrônica e Sistema Embarcados (LAESE) durante a disciplina de Microcontroladores no Instituto Federal do Ceará (IFCE).

O objetivo da atividade é utilizar o protocolo ESP-NOW para estabelecer comunicação entre diferentes circuitos com ESP32 sem a necessidade de conexão com rede WiFi. Nesse sentido, ambos circuitos devem ser capazes de enviar e receber mensagens de controle de um LED. Para isso foi feita a montagem do circuito e o desenvolvimento do código que realiza a operação proposta entre diferentes circuitos.

2. Materiais utilizados

Os materiais para a construção e acionamento do circuito incluem:

- 1 ESP32 30 pinos;
- 1 cabo micro USB;
- 1 protoboard;
- 1 LED;
- 1 buzzer;
- 1 botão;
- 2 resistores;
- 3 cabos jumper macho-fêmea;
- 2 cabos jumper macho-macho.

3. Montagem do circuito

Para realizar a montagem do circuito, é preciso seguir os seguintes passos:

- Ligar o ESP32 em uma fonte de alimentação com o cabo micro USB.
- Ligar o pino GND do ESP32 na região de alimentação da protoboard através de um jumper macho-fêmea.
- Posicionar o LED na região de componentes da protoboard.
- Posicionar um pino do resistor na região de alimentação e a outro em paralelo com o pino negativo do LED.
- Ligar o GPIO13 do ESP32 em paralelo com o pino positivo do LED através de um jumper macho-fêmea.
- Posicionar o buzzer na região de componentes da protoboard.
- Ligar o pino positivo do buzzer em paralelo com o pino positivo do LED através de um jumper macho-macho.
- Ligar o pino negativo do buzzer em paralelo com a região de alimentação da protoboard através de um jumper macho-macho.
- Posicionar o botão na região de componentes da protoboard.
- Posicionar uma extremidade do resistor em paralelo com um dos pinos do botão e a outra extremidade na região de alimentação da protoboard.
- Ligar um dos pinos do botão no GPIO12 do ESP32 através de um jumper macho-fêmea.

4. Implementação do código

O código responsável por controlar o LED através de um canal de comunicação com o ESP-NOW entre diferentes circuitos foi desenvolvido com a linguagem C++ no editor de código Visual Studio Code juntamente com sua extensão PlatformIO.

```
#include <Arduino.h>
#include <WiFi.h>
#include <esp_wifi.h>
#include <esp_now.h>

#define LED 13
#define BUTTON 12

uint8_t broadcastAddress1[] = {0xf0, 0x24, 0xf9, 0x44, 0x10, 0
    x40};

typedef struct data {
  bool isLedOn;
} data;

data dataSent;
data dataReceived;

esp_now_peer_info_t peerInfo;

bool isLedOn = true;
```

```
21
22 void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t
       status) {
23   char macStr[18];
24   Serial.print("Packet to: ");
25   snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02
       x",
26            mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3],
                mac_addr[4], mac_addr[5]);
27   Serial.print(macStr);
28   Serial.print(" send status:\t");
29   Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery
       Success" : "Delivery Fail");
30 }
31
32 void updateLedState(data dataReceived) {
33   if (dataReceived.isLedOn) {
34     digitalWrite(LED, HIGH);
35   }
36   else {
37     digitalWrite(LED, LOW);
38   }
39 }
40
41 void onDataReceived(const uint8_t * mac, const uint8_t *
       incomingData, int len) {
42   memcpy(&dataReceived, incomingData, sizeof(dataReceived));
43   Serial.print("Bytes received: ");
44   Serial.println(len);
45   Serial.print("Is Led On? ");
46   Serial.println(dataReceived.isLedOn);
47   Serial.println();
48   updateLedState(dataReceived);
49 }
50
51 void readButton() {
52   int buttonState = digitalRead(BUTTON);
53   if (buttonState == LOW) {
54     isLedOn = !isLedOn;
55     dataSent.isLedOn = isLedOn;
56   }
57 }
58
59 void checkDataSentSuccess(esp_err_t result) {
60   if (result == ESP_OK) {
61     Serial.println("Sent with success");
62   }
63   else {
64     Serial.println("Error sending the data");
```

```
  }
}

void readMacAddress(){
  uint8_t baseMac[6];
  esp_err_t ret = esp_wifi_get_mac(WIFI_IF_STA, baseMac);
  if (ret == ESP_OK) {
    Serial.printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                  baseMac[0], baseMac[1], baseMac[2],
                  baseMac[3], baseMac[4], baseMac[5]);
  } else {
    Serial.println("Failed to read MAC address");
  }
}

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

  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);

  WiFi.mode(WIFI_STA);

  Serial.print("ESP32 MAC Address: ");
  readMacAddress();

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_register_send_cb(onDataSent);

  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }

  esp_now_register_recv_cb(esp_now_recv_cb_t(onDataReceived));
}

void loop() {
  readButton();
  esp_err_t result = esp_now_send(0, (uint8_t *) &dataSent,
     sizeof(data));
```

```
113   checkDataSentSuccess(result);
114   delay(100);
115 }
```

5. Resultados

A montagem e implementação do código para estabelecer comunicação entre diferentes circuitos por meio do ESP-NOW foi concluída de forma bem sucedida. A seguir, pode-se visualizar a forma do circuito no final da prática, assim como os testes realizados e uma demonstração do seu funcionamento.

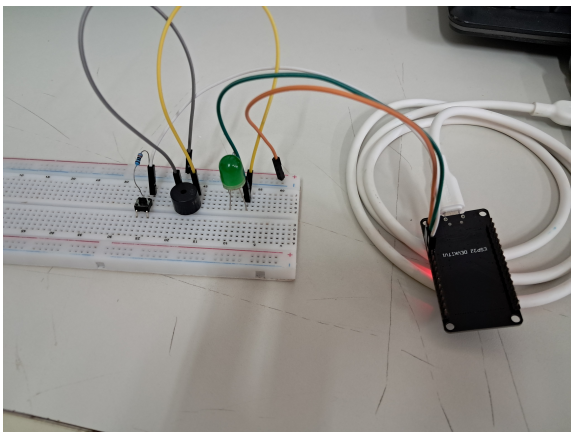

Figure 1. Circuito montado com um LED verde que é acionado e um botão que aciona outro circuito

O vídeo que demonstra o circuito em funcionamento está disponível em: Resultado da atividade prática 10

6. Conclusão

Após a finalização dessa atividade prática em laboratório, foi possível projetar o circuito e desenvolver o código que estabelece um canal de comunicação entre diferentes microcontroladores, além de controlar um LED por meio do protocolo ESP-NOW. Isso foi possível com o uso de um microcontrolador ESP32 e alguns componentes eletrônicos como uma protoboard, um LED verde, um buzzer, um botão, dois resistores e alguns cabos jumper. Assim, o objetivo proposto foi atingido com sucesso.