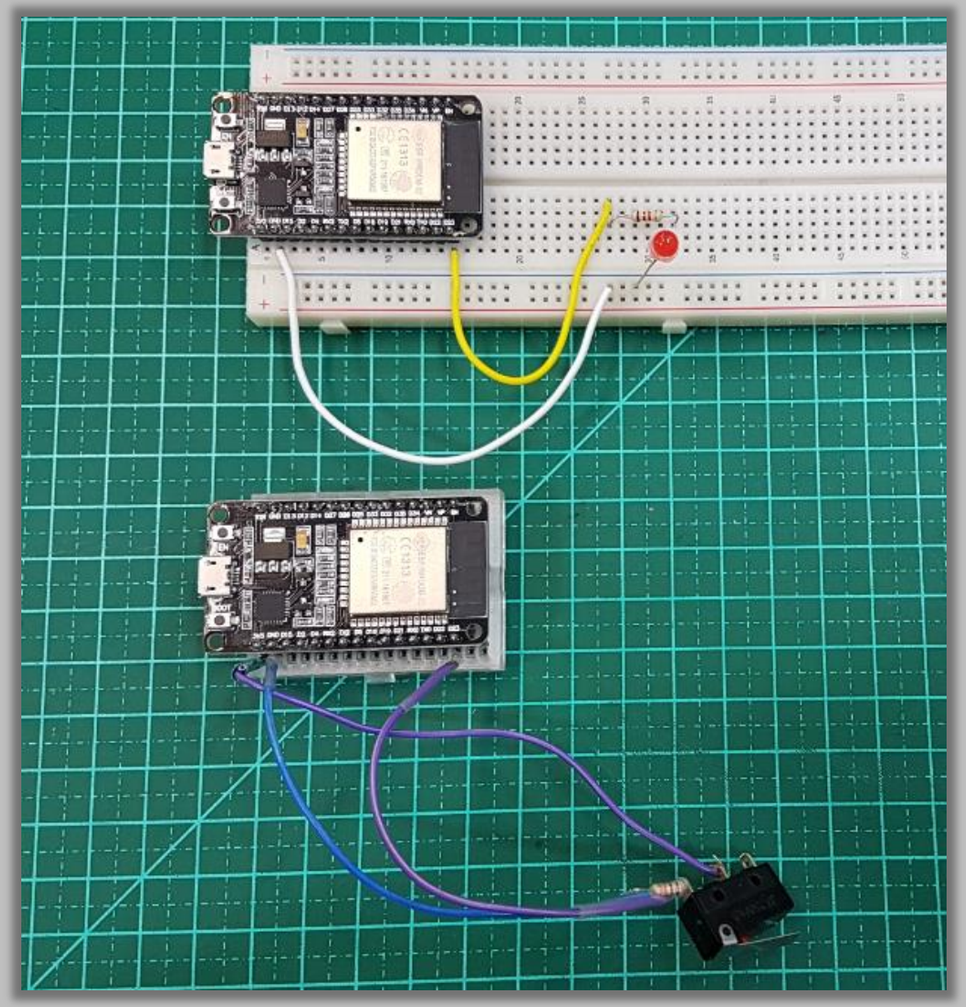
ESP32 Controle Remoto com Sockets

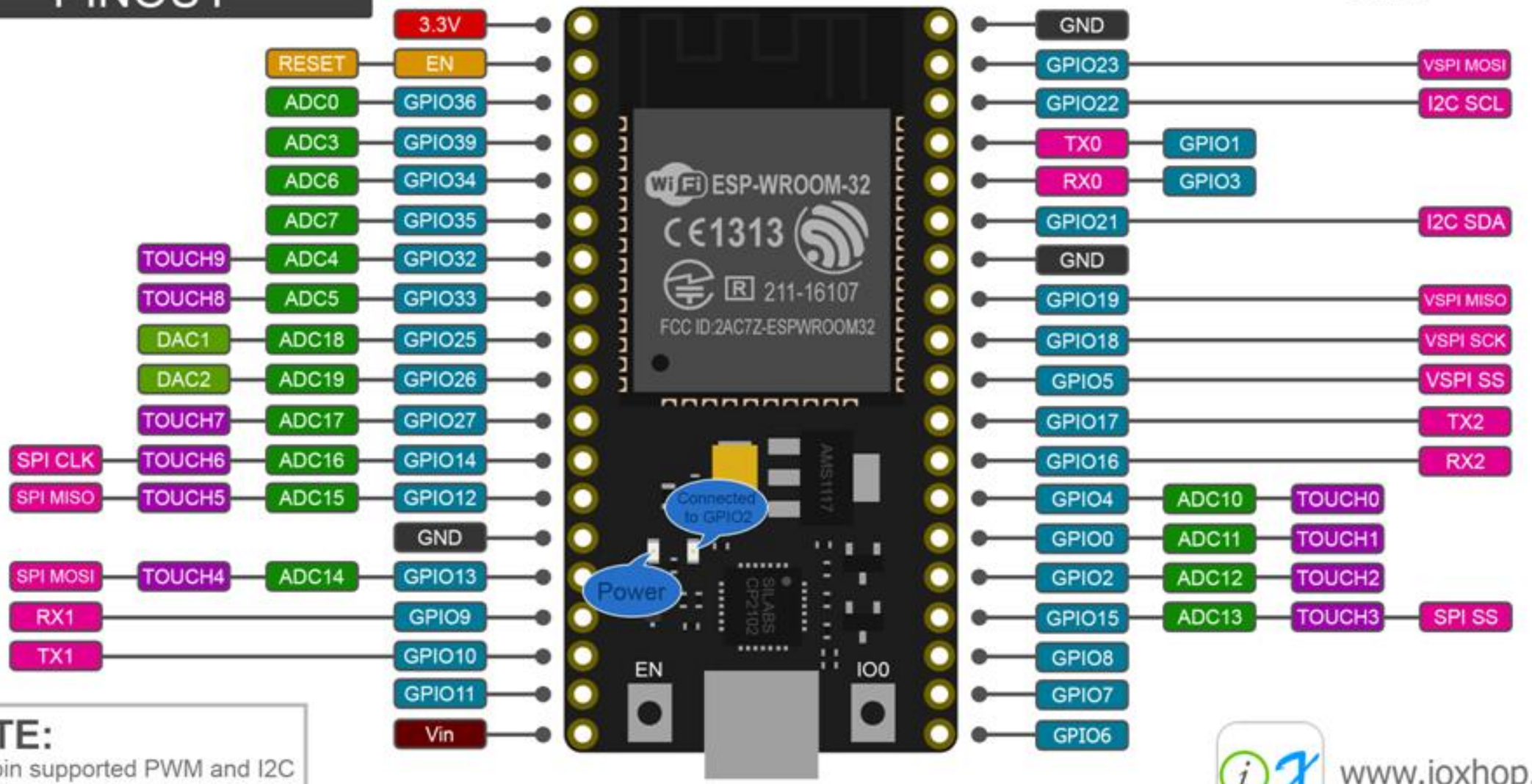


Por Fernando Koyanagi

HSMi

NodeMCU-32S

PINOUT



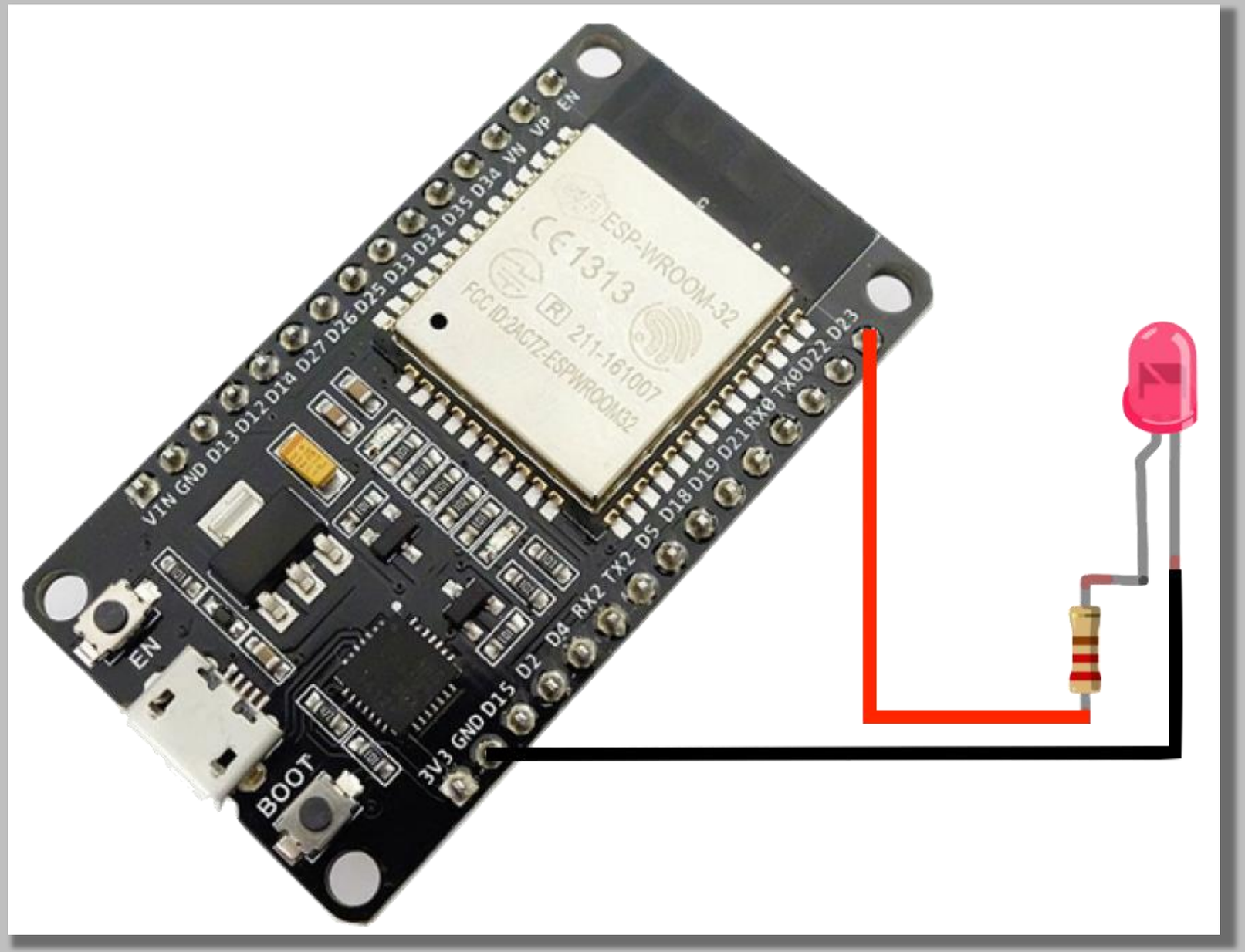
NOTE:

All pin supported PWM and I2C

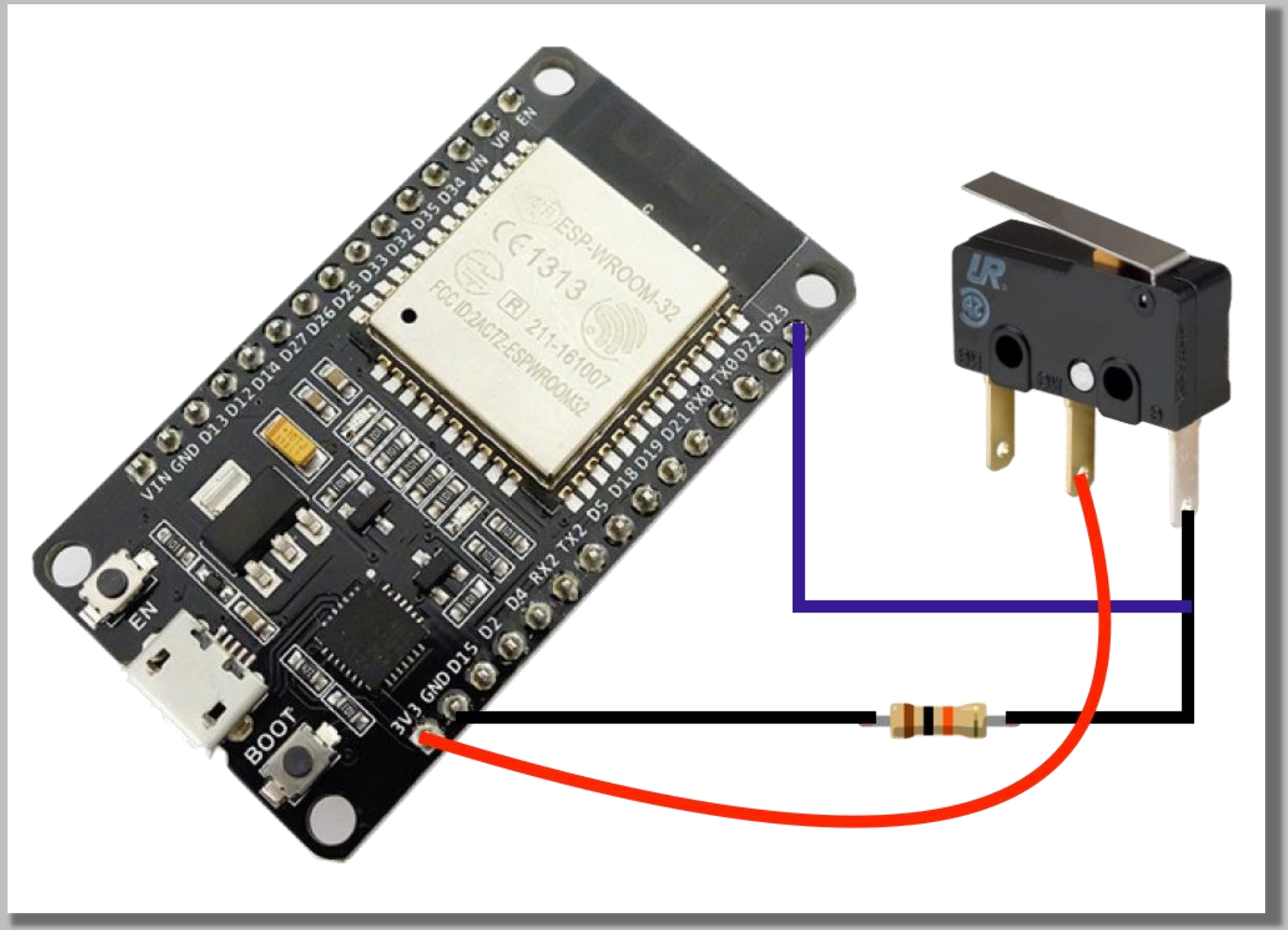
Pin current 6mA (Max. 12mA)

www.ioxhop.com

Montagem Server



Montagem Client



Em www.fernandok.com

Download arquivos INO do código fonte

ESP32Sockets.ino

```
//Arquivo principal. Neste arquivo vão os 'includes' e as configurações principais
//que são compartilhadas entre os outros arquivos .ino
#include <WiFi.h>

#define SSID "ESP32Server"
#define PASSWORD "87654321"
#define SERVER_PORT 5000

//Protocolo que o Server e o Client utilizarão para se comunicar
enum Protocol{
    PIN, //Pino que se deseja alterar o estado
    VALUE, //Estado para qual o pino deve ir (HIGH = 1 ou LOW = 0)
    BUFFER_SIZE //O tamanho do nosso protocolo. IMPORTANTE: deixar sempre como último do enum
};

//Diretiva de compilação que informará qual arquivo que queremos que seja compilado
//Caso queira que o arquivo Client.ino seja compilado, remova ou comente a linha do '#define'
//abaixo
//Caso queira que o arquivo Server.ino seja compilado, deixe o '#define IS_SERVER' abaixo descomentado
#define IS_SERVER
```

Server.ino

```
//Apenas vai compilar o código contido neste arquivo
//caso IS_SERVER esteja definido
#ifdef IS_SERVER

//Cria o server na porta definida por 'SERVER_PORT'
WiFiServer server(SERVER_PORT);

void setup()
{
    //Coloca este ESP como Access Point
    WiFi.mode(WIFI_AP);
    //SSID e Senha para se conectarem a este ESP
    WiFi.softAP(SSID, PASSWORD);
    //Inicia o server
    server.begin();
}
```

Server.ino

```
void loop()
{
    //Verifica se tem algum cliente se conectando
    WiFiClient client = server.available();
    if (client)
    {
        //Se o cliente tem dados que deseja nos enviar
        if (client.available())
        {//Criamos um buffer para colocar os dados
            uint8_t buffer[Protocol::BUFFER_SIZE];
            //Colocamos os dados enviados pelo cliente no buffer
            int len = client.read(buffer, Protocol::BUFFER_SIZE);
            //Verificamos qual o pino que o cliente enviou
            int pinNumber = buffer[Protocol::PIN];
            //Verificamos qual o valor deste pino
            int value = buffer[Protocol::VALUE];
            //Colocamos o pino em modo de saída
            pinMode(pinNumber, OUTPUT);
            //Alteramos o estado do pino para o valor passado
            digitalWrite(pinNumber, value);
        }

        //Fecha a conexão com o cliente
        client.stop();
    }
}
//Encerra o #ifdef do começo do arquivo
#endif
```

Client.ino

```
//Apenas vai compilar o código contido neste arquivo
//caso IS_SERVER NÃO esteja definido
//(if n def, atenção para o 'n')
#ifndef IS_SERVER

//Pino que vamos fazer a leitura
#define IN_PIN 23

void setup(){
    //Colocamos o pino em modo de leitura
    pinMode(IN_PIN, INPUT);
    //Conectamos Access Point criado
    //pelo outro ESP
    WiFi.begin(SSID, PASSWORD);

    //Esperamos conectar
    while (WiFi.status() != WL_CONNECTED){
        delay(500);
    }
}
```

Client.ino

```
void loop(){
    //Variável que utlizaremos para conectar ao servidor
    WiFiClient client;
    //Se não conseguiu se conectar então retornamos
    if (!client.connect(WiFi.gatewayIP(), SERVER_PORT)){
        return;
    }

    //Criamos um buffer para colocar os dados
    uint8_t buffer[Protocol::BUFFER_SIZE];
    //Fazemos a leitura do pino
    int value = digitalRead(IN_PIN);
    //Colocamos no buffer o número do pino
    //cujo estado queremos enviar
    buffer[Protocol::PIN] = IN_PIN;
    //Colocamos no buffer o estado atual do pino
    buffer[Protocol::VALUE] = value;
    //Enviamos e finalizamos a conexão
    client.write(buffer, Protocol::BUFFER_SIZE);
    client.flush();
    client.stop();
}
//Encerra o #ifndef do começo do arquivo
#endif
```

Em www.fernandok.com

Download arquivos INO do código fonte