Exemplo de como usar a placa Arduino Mega com WiFi (ESP8266) Integrado

Este texto apenas inclui passos a mais no tutorial referenciado abaixo, para facilitar ainda mais o uso da placa.

SMART KITS BLOG. Primeiros passos com Arduino Mega Wifi. 2024. Disponível em: <https://blog.smartkits.com.br/primeiros-passos-com-o-arduino-mega-wifi/>. Acesso em: 11 Out. 2024.

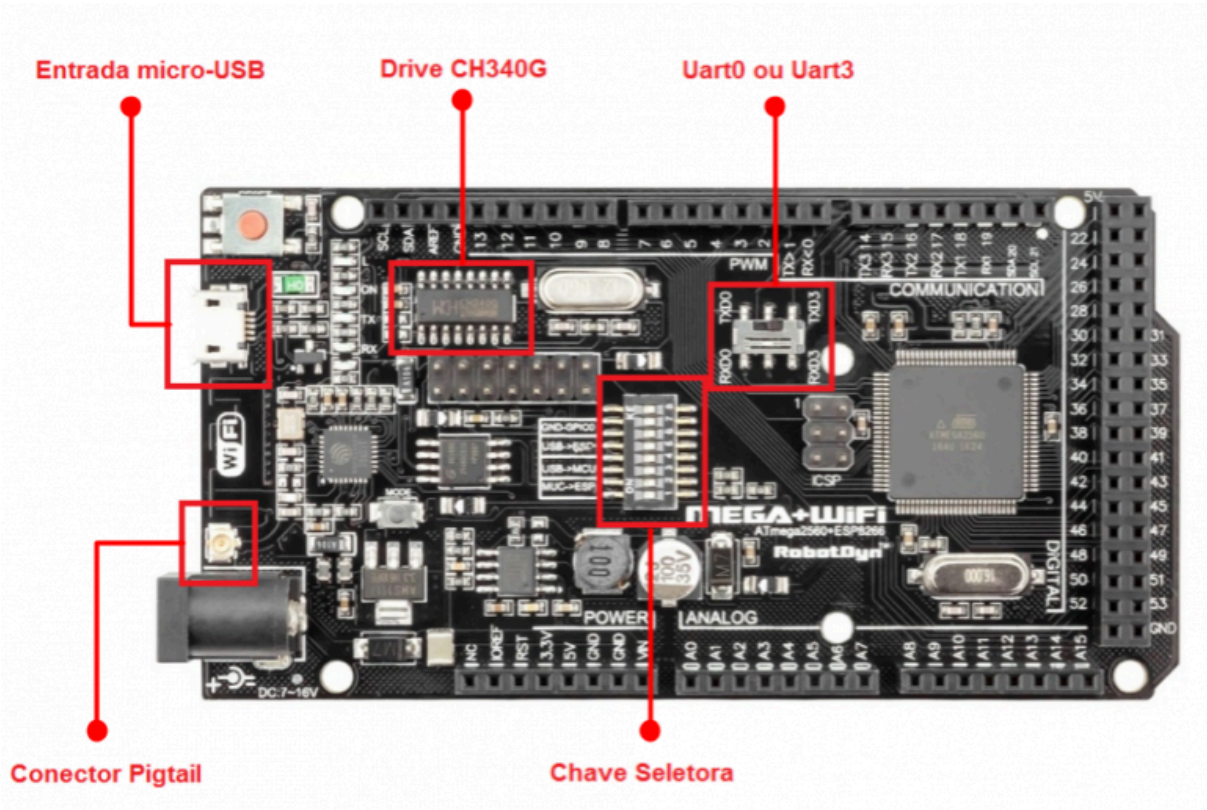

Figura 1 – Principais componentes da placa.

Exemplo de Servidor Web para controlar o LED integrado a placa

Montando o Projeto

Antes de cada etapa da programação devemos observar a posição dos DIP-switches, temos combinações diferentes para gravar o ESP8266, gravar o

ATmega2560 e para que o ESP8266 se comunique com o ATmega2560 pela UART3, que é a forma que utilizaremos nesse exemplo.

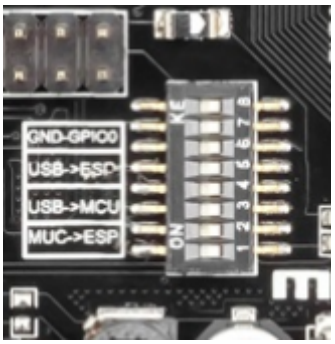

Figura 2 – DIP-switches de seleção do modo de operação da placa.

Logo abaixo temos a tabela com as combinações DIP Switches. Preste bastante atenção antes de cada etapa, para definir a configuração correta de funcionamento da placa.

	1	2	3	4	5	6	7	8
CH340 connect to ESP8266 (upload sketch)	OFF	OFF	OFF	OFF	ON	ON	ON	NoUSE
CH340 connect to ESP8266 (connect)	OFF	OFF	OFF	OFF	ON	ON	OFF	NoUSE
CH340 connect to ATmega2560 (upload sketch)	OFF	OFF	ON	ON	OFF	OFF	OFF	NoUSE
CH340 connect to Mega2560 COM3 connect to ESP8266	ON	ON	ON	ON	OFF	OFF	OFF	NoUSE
Mega2560+ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF	NoUSE
All modules work independent	OFF	OFF	OFF	OFF	OFF	OFF	OFF	NoUSE

Figura 3 – Tabela de modos de funcionamento da placa.

Também temos uma chave para selecionar a UART que o ATmega irá utilizar para se comunicar com o ESP8266.

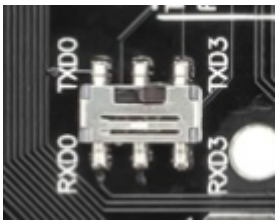

Figura 4 – Chave de seleção UART.

Para este exemplo usaremos apenas a UART3.

Programando o ESP8266

1. **Desconecte a placa do computador para configurar os DIP-switches;**
2. **Configure os DIP-switches da placa Mega WiFi no modo - CH340 connect to ESP8266 (upload sketch), conforme a figura 3.**

CH340 connect to ESP8266 (upload sketch)	OFF	OFF	OFF	OFF	ON	ON	ON	NoUSE

3. Depois vá até o menu de placas e selecione a placa ESPDuino (ESP-12 Module). Caso não funcione o upload do código, tente com a placa ESPDuino (ESP-13 Module).

   ATENÇÃO! Se as placas acima não estiverem na lista, procure na internet como adicionar as placas ESP8266 na IDE do Arduíno;

4. Selecione a porta COM correspondente a placa detectada pelo Windows.
5. Compile e carregue o código do arquivo LEDBuiltInMegaWifi_esp8266.ino do repositório abaixo, para o ESP8266, **lembrando de substituir o SSID e a senha de login da sua rede no código.**

   https://github.com/marcosvirgilio/ArduinoMegaWiFi_exemplo/

6. Se acontecer algum erro de carregamento do código, instale o driver CH341SER.EXE para as placas ESP8266 e revise se apareceu uma porta COM com número maior para seleção.

Programando o Mega2560

1. **Desconecte a placa do computador para configurar os DIP-switches;**
2. **Configure os DIP-switches da placa Mega WiFi no modo - CH340 connect to ATmega2560 (upload sketch), conforme a figura 3.**

CH340 connect to ATmega2560 (upload sketch)	OFF	OFF	ON	ON	OFF	OFF	OFF	NoUSE

3. Depois vá até o menu de placas e selecione a placa Arduino Mega or Mega 2560;
4. Selecione a porta COM correspondente a placa detectada pelo Windows.
5. Compile e carregue o código do arquivo LEDBuiltInMegaWifi_atmega.ino do repositório abaixo, para a placa;

https://github.com/marcosvirgilio/ArduinoMegaWiFi_exemplo/

Colocando para Funcionar

1. **Desconecte a placa do computador para configurar os DIP-switches;**
2. **Configure os DIP-switches da placa Mega WiFi no modo - CH340 connect to Mega2560 COM3 connect ESP8266, conforme a figura 3.**

CH340 connect to Mega2560 COM3 connect to ESP8266	ON	ON	ON	ON	OFF	OFF	OFF	NoUSE

3. Conecte a placa novamente ao computador;
4. **Abra o Monitor Serial e defina a velocidade do mesmo para 115200;**
5. Observe qual ip foi atribuído para a placa Mega WiFi. Acione o botão reset da placa caso não tenha aparecido nenhuma mensagem.

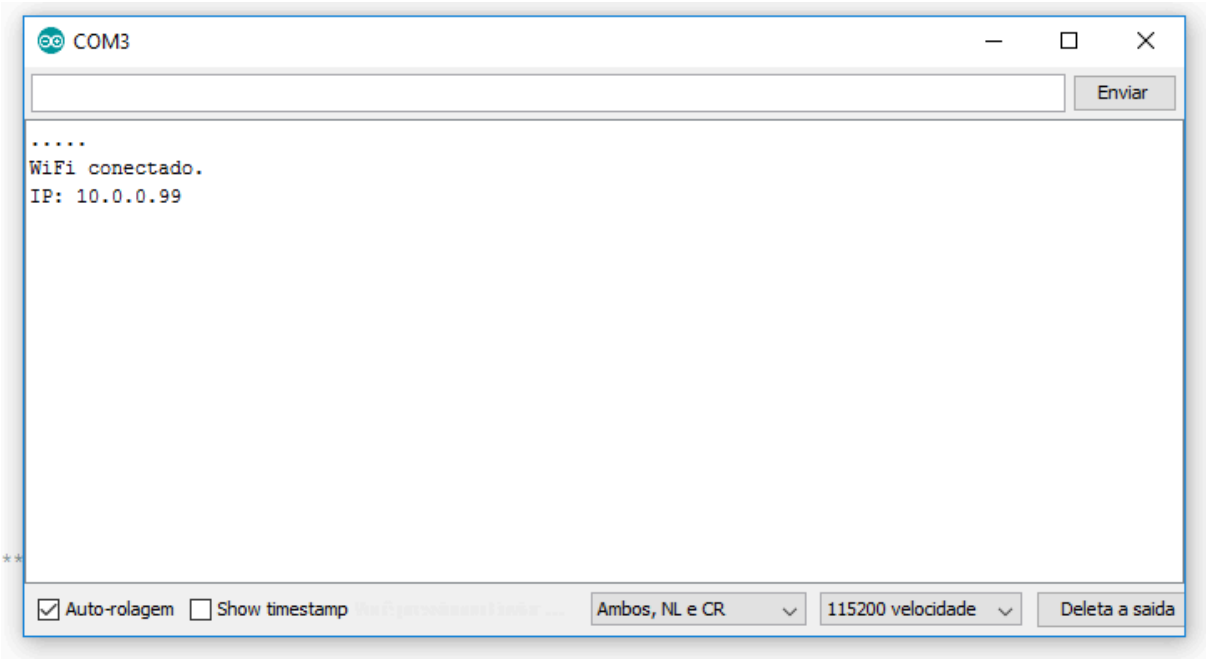

Figura 5 – Mensagens enviadas pelo Arduino Mega WiFi.

6. Abra o seu navegador e digite como endereço o ip informado no Serial Monitor. Se tudo estiver funcionando corretamente, uma página semelhante a da imagem abaixo será exibida.

Figura 6 – Pagina HTML.

Entendendo o código

Primeiramente vamos analisar as partes mais importantes do código carregado no ESP8266.

A primeira coisa que você precisa fazer é incluir o ESP8266WiFi biblioteca.

```
#include <ESP8266WiFi.h>
```

Troque o conteúdo das variáveis ssid e password pelas respectivas ssid e senha da sua rede WiFi.

```
// Substitua pelas suas credenciais de rede.
const char* ssid     = "sua-rede";
const char* password = "sua-senha";
```

Em seguida, você define seu servidor da Web para a porta 80.

```
// Defina um servidor com a porta 80.
WiFiServer server(80);
```

Dentro da função setup() configure a porta serial para 115200 de baudrate,que será utilizada para enviar informações para o ATmega2560, como o IP por exemplo.

```
Serial.begin(115200);
```

Em seguida o ESP8266 irá tenta se conectar a rede com as credenciais definidas acima. E enviará as informações via serial ao ATmega2560.

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi conectado.");
Serial.print("IP: ");
Serial.println(WiFi.localIP());
```

Inicie o servidor web.

```
 //Inicie o servidor.
server.begin();
```

Na função loop() serão tratadas as solicitações dos clientes que se conectarem ao nosso servidor web.

O ESP8266 está sempre ouvindo os clientes conectados com esta linha:

```
WiFiClient client = server.available(); // Escute os clientes conectados
```

Quando uma solicitação é recebida de um cliente, salvaremos os dados recebidos. O loop while a seguir estará em execução enquanto o cliente permanecer conectado.

```
while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop enquanto cliente estiver conectado.

      currentTime = millis();

      if (client.available()) {                // Se holver bytes para ler do cliente,

        char c = client.read();                // faça a leitura.

        header += c;
```

```
      //Outras tarefas ...
}
```

Após fazer a leitura de toda solicitação recebida do cliente, podemos verificar qual URL está sendo recebida, pois ela irá mudar dependendo do comando que usamos. Arbitrei que “/13/on” será usada para ativar e “/13/off” para desativar. Após verificar qual dessas opções foi recebida, envie uma mensagem correspondente via serial ao ATmega2560.

```
// Procure o trecho "GET /13/on" dentro da solicitação do recebida do cliente.
            if (header.indexOf("GET /13/on") >= 0) {
              //Envie um comando para Mega2560 via serial.
              Serial.println("LED_ON");
              //Altere a variavel de estado.
              outputState = "Ligado";
            } else if (header.indexOf("GET /13/off") >= 0) {
              //Envie um comando para Mega2560 via serial.
              Serial.println("LED_OFF");
              //Altere a variavel de estado.
              outputState = "Desligado";
            }
```

Logo após enviaremos uma página de resposta ao cliente.

```
// Pagina HTML
              client.println("<!DOCTYPE html><html>");
              client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
              client.println("<link rel=\"icon\" href=\"data:,\">");
              // CSS para estilizar a pagina
              client.println("<style>html { font-family: Helvetica; display:
inline-block; margin: 0px auto; text-align: center;}");
              client.println("h1,p {font-weight: bold;color: #126e54; font-size:
32px;}");
              client.println("p {font-size: 16px;}");
              client.println(".button { background-color: #1BAE85; border: none;
color: white; padding: 16px 40px;");
              client.println("text-decoration: none; font-size: 30px; margin:
2px; cursor: pointer;}");
              client.println("</style></head>");

              client.println("<body><h1>Arduino Mega WiFi Web Server</h1>");

              // Mostre o estado atual do pino 13, aqui representado pela
variavel de estado outputState.
              client.println("<p>Pino 13 - Estado " + outputState + "</p>");
              // Se outputState estiver como Desligado, crie um botao com texto
Ligar.
              if (outputState == "Desligado") {
                client.println("<p><a href=\"/13/on\"><button
class=\"button\">Ligar</button></a></p>");
              } else {
                // Se outputState estiver como Ligado, crie um botao com texto
Desligar.
                client.println("<p><a href=\"/13/off\"><button class=\"button
button2\">Desligar</button></a></p>");
              }
              client.println("</body></html>");
              // A resposta HTTP termina com outra linha em branco.
```

```
              client.println();
              break;
            } else { // Se você recebeu uma nova linha, limpe currentLine
              currentLine = "";
            }
          } else if (c != '\r') {  // Se você tiver mais alguma coisa além de um caractere de retorno de carro,
            currentLine += c;      // Adicione-o ao final do currentLine.
          }
        }
      }
      // Limpe a variável de cabeçalho
      header = "";
      // Fece a conexao.
      client.stop();
```

Vale destacar que a página será ligeiramente diferente dependendo da solicitação do cliente.Por exemplo se o cliente mandou o comando para ativar a saída do Arduino, enviaremos uma página com um botão escrito “Desligar” com um link para “/13/off” para que o cliente tenha somente a opção de desligar.

```
// Mostre o estado atual do pino 13, aqui representado pela variavel de estado outputState.
              client.println("<p>Pino 13 - Estado " + outputState + "</p>");
              // Se outputState estiver como Desligado, crie um botao com texto Ligar.
              if (outputState == "Desligado") {
                client.println("<p><a href=\"/13/on\"><button class=\"button\">Ligar</button></a></p>");
              } else {
                // Se outputState estiver como Ligado, crie um botao com texto Desligar.
                client.println("<p><a href=\"/13/off\"><button class=\"button button2\">Desligar</button></a></p>");
              }
```

Agora vamos dar uma olhada no código para o ATmega2560. Primeiramente definimos uma string que irá armazenar as mensagens recebidas do ESP8266.

```
String msg;//String para armazenar a mensagem recebida pela porta serial 3.
```

Configuramos o pino 13 como saída digital, aqui usamos a definição LED_BUILTIN pois faz referência ao pino que controla o LED integrado da placa, geralmente o 13.

```
//Defina o pino LED_BUILTIN (13), como saida.
  pinMode(LED_BUILTIN, OUTPUT);
```

Definimos as duas portas seriais que serão utilizadas, a Serial3 está conectada ao ESP8266 e a Serial simples está conectada ao CH340G.

```
//Defina as porta serial para comunicação usb.
  Serial.begin(115200);
  //Defina as porta serial para comunicação com ESP8266.
  Serial3.begin(115200);
```

Dentro da função loop() o Mega2560 ficará sempre verificando o buffer da UART3, se existir algum dado, este será lido e replicado para a UART0 e enviado via USB para o usuário.

```
//Leitura de um byte.
char data = Serial3.read();
//Imprima o mesmo dado pela porta usb.
Serial.print(data);
```

Acrescente cada caracter recebido a nossa string *msg.*

```
//Acrescente o caractere recebido a string de mensagem.
msg += data;
```

Faça uma busca pelas palavras LED_ON ou LED_OFF. Dependendo do resultado ative ou desative a saida 13.

```
if (msg.indexOf("LED_ON") > 0) {//Verifica a ocorrencia do trecho "LED_ON" na mensagem recebida.

      //Ative a saida.

      digitalWrite(LED_BUILTIN, HIGH);

    }

```

```
6     else if (msg.indexOf("LED_OFF")> 0) {//Verifica a ocorrencia do trecho
"LED_OFF" na mensagem recebida.
7
        //Desligue a saida.
8
        digitalWrite(LED_BUILTIN, LOW);

      }
```

Bom por hoje é tudo. Se interessou pelo modelo Mega WiFi? Visite o site da loja e adquira o seu Arduino Mega WiFi hoje mesmo.

Escrito por

Yure Albuquerque

Graduando em Tecnologia Mecatrônica Industrial. Tenho experiência com Arduino e Raspberry Pi. Atualmente experimentando novas tecnologias como ESP32 e compartilhando no blog da Smart Kits.