**Sistemas de Comunicação Móvel, 1º Trabalho de Laboratório:**

## Problemas encontrados e soluções:

- Aquando da tentativa de comunicação entre os dois Arduinos, notou-se que a receção de mensagens num dos lados só acontecia após terminar a conexão. Verificou-se que este problema se devia ao facto de que, ao verificar a existência de bytes na leitura, usava-se a seguinte linha de código:

  ```
  if (client.available() == 0)
  ```

  O que se notou foi que a adição do "`== 0`" impedia a leitura do buffer, pelo que ao retirar esta porção o problema foi resolvido.

- Também foi notório alguns problemas a conseguir ligar/desligar os LEDs, uma vez que quando se enviava os comandos `ON/OFF` para ligar/desligar a luz, obtínhamos input inválido. Para resolver este problema, adicionámos à comparação o símbolo `\r`, o que finalmente permitiu obter comparações válidas e assim alterar o estado do LED.

## Código fonte devidamente comentado:

```
#include <ESP8266WiFi.h>

#ifndef STASSID
#define STASSID "Galaxy A52 5G863E"
#define STAPSK  "fpww0659"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

WiFiServer server(25);

void setup() {
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, 0);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print(F("Connecting to "));
  Serial.println(ssid);

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

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();
  Serial.println(F("WiFi connected"));
```

```
  // Start the server
  server.begin();
  Serial.println(F("Server started"));

  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {
  String str, variant;
  int val;

  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  Serial.println(F("new client"));
  Serial.print(F("802.11 Variant/Channel is: "));
  WiFiPhyMode_t mode = WiFi.getPhyMode();
  if (mode == WIFI_PHY_MODE_11B){
    variant = "802.11 b";
  } else if (mode == WIFI_PHY_MODE_11N){
    variant = "802.11 n";
  } else if (mode == WIFI_PHY_MODE_11G){
    variant = "802.11 g";
  } else {
    variant = "Invalid variant";
  }
  Serial.println(variant + F("/") + WiFi.channel());
  Serial.print(F("RSSI of the used SSID: "));
  Serial.println(WiFi.RSSI());

  client.setTimeout(5000); // timeout for client

  while (client.connected()) {
    // Check if there are any bytes available for read
    if (client.available()){
      String req = client.readStringUntil('\n');
      Serial.println(F("request: "));
      Serial.println(req);

      // Check if the request is for ON/OFF
      if (req.equals("OFF\r")) {
        val = 1;
      } else if (req.equals("ON\r")) {
        val = 0;
      } else {
        Serial.println(F("invalid request"));
        val = digitalRead(LED_BUILTIN);
      }

      // Set LED according to the request
      digitalWrite(LED_BUILTIN, val);
    } else {
      // Check if there are any inputs from Serial
      if (Serial.available()>0){
        str = Serial.readStringUntil('\n');
        client.println(str);
      }
    }
  }

  // The client will be *flushed* then disconnected
  // when the function returns and 'client' object is destroyed (out-of-scope)
  // flush = ensure written data are received by the other side
  Serial.println(F("Disconnecting from client"));
}
```