Inhaltsverzeichnis

Fritzbox Call Monitor with ESP8266

At start it was just a question of a relevant, if it could not be indicated somehow, that there's a phone call ongoing on the Fritzbox.

When thinking about it, the first idea was to do something with the TR-064 protocol, but in the end it's much more simple:

When activated with the special phone number

```
#96*5*
```

the Fritzbox reports SIP information on port 1012 like this

```
20.02.16 18:28:24;RING;0;017xxxxxxxx;8xxxxxx;SIP1;
20.02.16 18:28:27;DISCONNECT;0;0;
20.02.16 18:29:03;CALL;1;12;9xxxxx;053xxxxxx;SIP2;
20.02.16 18:29:31;CONNECT;1;12;053xxxxxx;
20.02.16 18:29:52;DISCONNECT;1;21;
20.02.16 18:30:00;CALL;1;12;9xxxxx;053xxxxxx;SIP2;
20.02.16 18:30:18;CONNECT;1;12;053xxxxxx;
20.02.16 18:51:39;DISCONNECT;1;1281;
20.02.16 18:58:12;CALL;1;11;9xxxxx;041xxxxxxxxx;SIP2;
20.02.16 18:58:26;DISCONNECT;1;0;
```

This information can be viewed under linux simply with

```
netcat fritz.box 1012
```

Implementation on a ESP8266

As a quick and dirty solution, I took a TCP client sample and modified it to connect first to Wifi, then to the Fritzbox and then listen to the incoming status messages.

Whenever a CONNECT or DISCONNECT is discovered, the build-in LED is switched on or off.

Job done...

ESP8266_Fritzbox_Callmonitor.ino

```
#include <ESP8266WiFi.h>

const char* ssid = "yourWiFiSSID";
const char* password = "yourpassword";

const char* host = "192.168.2.1";
```

```
const char *connect = ";CONNECT";
const char *disconnect = ";DISCONNECT";

// for Wemos D1
#if 0
  #define OUTPUT_PIN (2)
  #define OUT_ON LOW
  #define OUT_OFF HIGH
#else
 // for OBI Socket 2
  #define OUTPUT_PIN (4)
  #define OUT_ON HIGH
  #define OUT_OFF LOW
#endif


void setup()
{

pinMode(OUTPUT_PIN, OUTPUT);     // Initialize GPIO pin as an output

  Serial.begin(115200);
  Serial.println();

  Serial.printf("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");
  digitalWrite(OUTPUT_PIN, OUT_ON);   // Turn the LED on
  delay(500);              // Wait for 500 ms
  digitalWrite(OUTPUT_PIN, OUT_OFF);   // Turn the LED off


}


void loop()
{
  WiFiClient client;

  Serial.printf("\n[Connecting to %s ... ", host);
  if (client.connect(host, 1012))
  {
    Serial.println("connected]");

/*
```

```
    Serial.println("[Sending a request]");
    client.print(String("GET /") + " HTTP/1.1\r\n" +
                 "Host: " + host + "\r\n" +
                 "Connection: close\r\n" +
                 "\r\n"
                );

    Serial.println("[Response:]");
*/
    while (client.connected() || client.available())
    {
      if (client.available())
      {
        String line = client.readStringUntil('\n');
        Serial.println(line);
        if( line.indexOf( connect ) > -1) {
         Serial.println("line busy]");
          digitalWrite(OUTPUT_PIN, OUT_ON);   // Turn the OUTPUT on
        }
        if( line.indexOf( disconnect ) > -1) {
         Serial.println("line free]");
          digitalWrite(OUTPUT_PIN, OUT_OFF);   // Turn the OUTPUT off
        }
      }
    }
    client.stop();
    Serial.println("\n[Disconnected]");
  }
  else
  {
    Serial.println("connection failed!]");
    client.stop();
  }
  delay(5000);
}
```