Part 04
-
WiFi

Version: 2022-05-08

WiFi setup

It took me some time to get all pieces of the puzzle hooked up but I got it in the end.

Here is the frame to set-up WiFi with all bitsand bytes.

Note: only pieces of the code required are shown here. So this is not a fully operational sketch as such.

```
//------------------------------------------------------------------------
// includes
//------------------------------------------------------------------------
#include <WiFi.h>


//------------------------------------------------------------------------
// constants
//------------------------------------------------------------------------
//WiFi credentials
const char* wifiSSID           = "SSID";
const char* wifiPassword       = "WiFi-Password";
// hostname to be shown on the network like in DHCP, DNS, ...
const char* hostname           = "hostname";

//------------------------------------------------------------------------
// defines
//------------------------------------------------------------------------
// uncomment to activate serial logging
//#define DEBUG

//------------------------------------------------------------------------
// global variables
//------------------------------------------------------------------------
// to hold MAC address
String wifiMACAddress = WiFi.macAddress();

//------------------------------------------------------------------------
// Functions
//------------------------------------------------------------------------

//************************************************************************
void wifiConnect()
{
  // needed to allow setting hostname
  WiFi.config(INADDR_NONE, INADDR_NONE, INADDR_NONE, INADDR_NONE);
  // set hostname
  WiFi.setHostname(hostname);
  // act as a WiFi station/device
  WiFi.mode(WIFI_STA);
  // connect
  WiFi.begin(wifiSSID, wifiPassword);

  #if defined(DEBUG)
    dt = LocalTime();
    Serial.print(dt); Serial.print(" - WiFi attempting connection to ");
    Serial.println(wifiSSID);
  #endif

  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED)
  {
    #if defined(DEBUG)
      Serial.print('.');
```

```
    #endif
    // delay non-blocking
    time_now = millis();
    while(millis() < time_now + 500){}
    if ((++i % 16) == 0)
    {
      #if defined(DEBUG)
        Serial.println("");
        dt = LocalTime();
        Serial.print(dt); Serial.println(" - still trying to connect");
      #endif
    }

    // if too many retries, reboot
    if (i > 32)
    {
      dt = LocalTime();
      Serial.print(dt); Serial.println(" - Restarting...");
      ESP.restart();
    }
  }

  #if defined(DEBUG)
    Serial.println("");
  #endif
  dt = LocalTime();
  Serial.print(dt); Serial.println(" - WiFi connected");
  #if defined(DEBUG)
    Serial.print(dt); Serial.print(" -    MAC Address: ");
    Serial.println(wifiMACAddress);
    Serial.print(dt); Serial.print(" -    IP Address : ");
    Serial.println(WiFi.localIP());
    Serial.print(dt); Serial.print(" -    Hostname   : ");
    Serial.println(WiFi.getHostname());
    Serial.print(dt); Serial.println("");
  #endif
}
//**************************************************************************

//**************************************************************************
void wifiEvent(WiFiEvent_t event)
{
  #if defined(DEBUG)
    dt = LocalTime();
    Serial.print(dt); Serial.printf(" - [WiFi-event : %d] = ", event);

    switch (event)
    {
      case SYSTEM_EVENT_WIFI_READY:
        Serial.println("Interface ready");
        break;

      case SYSTEM_EVENT_SCAN_DONE:
        Serial.println("Completed scan for APs");
        break;

      case SYSTEM_EVENT_STA_START:
        Serial.println("STA Client started");
        break;

      case SYSTEM_EVENT_STA_STOP:
        Serial.println("STA Client stopped");
        break;
```

```
case SYSTEM_EVENT_STA_CONNECTED:
  Serial.println("Connected to AP");
  break;

case SYSTEM_EVENT_STA_DISCONNECTED:
  Serial.println("Disconnected from AP");
  break;

case SYSTEM_EVENT_STA_AUTHMODE_CHANGE:
  Serial.println("Auth changed");
  break;

case SYSTEM_EVENT_STA_GOT_IP:
  Serial.print("Obtained IP: ");
  Serial.println(WiFi.localIP());
  break;

case SYSTEM_EVENT_STA_LOST_IP:
  Serial.println("Lost IP");
  break;

case SYSTEM_EVENT_STA_WPS_ER_SUCCESS:
  Serial.println("WPS succeeded");
  break;

case SYSTEM_EVENT_STA_WPS_ER_FAILED:
  Serial.println("WPS failed");
  break;

case SYSTEM_EVENT_STA_WPS_ER_TIMEOUT:
  Serial.println("WPS timeout");
  break;

case SYSTEM_EVENT_STA_WPS_ER_PIN:
  Serial.println("WPS pin code");
  break;

case SYSTEM_EVENT_AP_START:
  Serial.println("AP started");
  break;

case SYSTEM_EVENT_AP_STOP:
  Serial.println("AP stopped");
  break;

case SYSTEM_EVENT_AP_STACONNECTED:
  Serial.println("AP Client connected");
  break;

case SYSTEM_EVENT_AP_STADISCONNECTED:
  Serial.println("AP Client disconnected");
  break;

case SYSTEM_EVENT_AP_STAIPASSIGNED:
  Serial.println("Assigned IP to client");
  break;

case SYSTEM_EVENT_AP_PROBEREQRECVED:
  Serial.println("Received probe request");
  break;

case SYSTEM_EVENT_GOT_IP6:
  Serial.println("IPv6 preferred");
  break;
```

```
      case SYSTEM_EVENT_ETH_START:
        Serial.println("Ethernet started");
        break;

      case SYSTEM_EVENT_ETH_STOP:
        Serial.println("Ethernet stopped");
        break;

      case SYSTEM_EVENT_ETH_CONNECTED:
        Serial.println("Ethernet connected");
        break;

      case SYSTEM_EVENT_ETH_DISCONNECTED:
        Serial.println("Ethernet disconnected");
        break;

      case SYSTEM_EVENT_ETH_GOT_IP:
        Serial.println("Obtained IP");
        break;

      default:
          break;
    }
  #endif
}
//**************************************************************************


//==========================================================================
// Setup
//==========================================================================
void setup()
{
  Serial.begin(115200);
  Serial.println("");Serial.println("");
  Serial.println("======================");
  Serial.print("Booting (Version: "); Serial.print(VERSION);
  Serial.println(")");
  Serial.println("======================");

  //--------------------------------------------------------------------------
  // WiFi setup section
  //--------------------------------------------------------------------------
  // Connect to WiFi network
  #if defined(DEBUG)
    WiFi.onEvent(wifiEvent);
  #endif
  wifiConnect();
  //--------------------------------------------------------------------------

  //--------------------------------------------------------------------------
  Serial.println("Ready");

}
//==========================================================================

//##########################################################################
// Main loop
//##########################################################################
void loop()
{

}
```

ESP32 – Connect to best WiFi from multiple options

Generally, we specify one SSID and one password for the ESP32 WiFi connection. However, what if you have credentials available for multiple WiFi sources around ESP32 and wish to connect to the best network available? ESP32 Arduino has a built-in library specifically for this purpose: ***WiFiMulti***.
Here are the steps required to add multiple SSID and password options to your sketch

- Import WiFiMulti along with WiFi

```
#include <WiFi.h>
#include <WiFiMulti.h>
```

- Create an object of type WiFiMulti

```
WiFiMulti wifiMulti;
```

- Add the multiple SSIDs and passwords using the `.addAP()` function in the setup

```
wifiMulti.addAP("SSID1", "password1");
wifiMulti.addAP("SSID2", "password2");
wifiMulti.addAP("SSID3", "password3");
```

- Connect to the WiFi using the `.run()` function.
  It returns the status of the WiFi connection. If connected, you can print the SSID it is connected to.

```
if(wifiMulti.run() == WL_CONNECTED)
{
  Serial.println(WiFi.SSID());
}
```

  The `.run()` function first checks if the WiFi is already connected to one of the options provided by you. If yes, it returns. Otherwise, it tries to connect to that SSID from the list which has the best RSSI.

You can try experimenting with two networks, keeping the ESP32 closer to one and then the other. The ESP32 will connect to different networks in both cases.
An example sketch is provided with the ESP32 Arduino Core.

```
/*
 *  This sketch trys to Connect to the best AP based on a given list
 *
 */

#include <WiFi.h>
#include <WiFiMulti.h>

WiFiMulti wifiMulti;

void setup()
{
  Serial.begin(115200);
  delay(100);

  wifiMulti.addAP("ssid_from_AP_1", "your_password_for_AP_1");
  wifiMulti.addAP("ssid_from_AP_2", "your_password_for_AP_2");
  wifiMulti.addAP("ssid_from_AP_3", "your_password_for_AP_3");

  Serial.print("Connecting to Wifi...");
  if(wifiMulti.run() == WL_CONNECTED)
  {
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address: "); Serial.println(WiFi.localIP());
  }
}

void loop()
{
  if(wifiMulti.run() != WL_CONNECTED)
  {
    Serial.println("WiFi not connected!");
    delay(1000);
  }
}
```

That’s it. You now have multiple options for WiFi connection and your ESP32 can connect to another available network if one network goes down. This makes the overall system more robust.