Solución Actividad 2

Una vez instalado el SDK de Moddable trata de realizar los siguientes programas que manejan la Wifi y la conexión de red.

1. **Hacer un programa que muestre por la serial información básica de los puntos de acceso (AP) disponibles.**

Tendremos que hacer uso de la clase `Wifi`. Esta clase está dentro del módulo `net`, por lo que en el `manifest.json` debe incluirse este:

```
1 {
2   "include": [
3     "$(MODDABLE)/examples/manifest_base.json",
4     "$(MODDABLE)/examples/manifest_net.json"
5   ],
6   "modules": {
7     "*": [
8       "./main"
9     ]
10  }
11 }
```

En el programa principal bastará con invocar el método `scan()` con una función callback, que se invocará cada vez que se encuentre un AP, que se encargue de mostrar la información:

```
1 import WiFi from "wifi";
2
3 WiFi.scan({}, ap => {
4
5     if (!ap) {
6         trace(`Ya estan todos los AP\n`);
7         return;
8     }
9
10    trace(`======== \n`);
11    trace(`Encontrado ${ap.ssid}\n`);
12    trace(`RSSI: ${ap.rssi}\n`);
13    trace(`autenticación: ${ap.authentication}\n`);
14
15 });
```

Como el parámetro ap estará a `null` cuando no haya más AP, en la línea 5 detectamos esa situación y sacamos mensaje correspondiente.

2. **Hacer programa que conecte a un AP, dado su identificador (ssid) y clave de acceso, pero sin hacerse cargo de las desconexiones.**

Si queremos conectarnos a un AP sin estar pendiente de las desconexiones, la clase Net se hace cargo de todo el proceso durante se inicialización. Basta, entonces con importar el módulo:

```
1 import Net from "net";
2
3 trace(`IP Address: ${Net.get("IP")}\n`);
4 trace(`MAC Address: ${Net.get("MAC")}\n`);
5 trace(`SSID: ${Net.get("SSID")}\n`);
6 trace(`BSSID: ${Net.get("BSSID")}\n`);
7 trace(`RSSI: ${Net.get("RSSI")}\n`);
```

La única cuestión es como hacerle llegar el ssid y la clave de acceso. Hay dos soluciones: la primera a través de la línea de comandos:

```
mcconfig -d -m -p esp32 ssid="MiWiFi" password="secreto"
```

La otra opción, por lo general más cómoda, es indicar estos parámetros en la sección `config` del `manifest.json`:

```
 1 {
 2   "include": [
 3     "$(MODDABLE)/examples/manifest_base.json",
 4     "$(MODDABLE)/examples/manifest_net.json"
 5   ],
 6   "modules": {
 7     "*": [
 8       "./main"
 9     ],
10     "config": {
11       "ssid": "MiWiFi",
12       "password": "secreto"
13     }
14   }
15 }
```

3. **Hacer programa que conecte a un AP, dado su identificador (ssid) y clave de acceso, y que se reconecte si se pierde la conexión.**

Si queremos que el ESP32 se reconecte al AP en caso de que la conexión se pierda, es necesario hacer en el programa principal la gestión completa de la conexión al AP a través de la clase `WiFi`:

```
 1 import config from "mc/config";
 2 import WiFi from "wifi";
 3 import Net from "net";
 4 import Timer from "timer";
 5
 6 const {ssid, password} = config;
 7
 8 WiFi.mode = 1; // station mode
 9
10 trace(`ssid: ${ssid}, pass: ${password}\n`);
```

```
11
12 const opcionesWifi = {ssid, password};
13
14 const monitor = new WiFi(opcionesWifi, msg => {
15   switch (msg) {
16     case WiFi.connected:
17       trace(`Conectado wifi, esperando IP\n`);
18       break; // still waiting for IP address
19     case WiFi.gotIP:
20       trace(`IP Address: ${Net.get("IP")}\n`);
21       trace(`MAC Address: ${Net.get("MAC")}\n`);
22       trace(`SSID: ${Net.get("SSID")}\n`);
23       trace(`BSSID: ${Net.get("BSSID")}\n`);
24       trace(`RSSI: ${Net.get("RSSI")}\n`);
25       break;
26     case WiFi.disconnected:
27       trace(`Wifi desconectada\n`);
28       Timer.set(function() {
29         trace(`Vamos a tratar de reconectarnos\n`);
30         WiFi.connect(opcionesWifi);
31       }, 5 * 1000);
32
33       break;  // connection lost
34   }
35 });
```

Al instanciar el objeto WiFi pasamos un callback que recibirá un mensaje (`msg`) indicando el estado de la conexión:

- `WiFi.connected` → si lla conectó al AP y está pendiente de la IP.
- `WiFi.gotIP` → ya está la conexión completa y pondemos consultar todos los parámetros
- `WiFi.disconnected` → se ha perdido la conexión. En ese caso volvemos a intentar la conexión, pero no inmediatamente, sino después de 5 segundos. Para ello usamos un temporizador (líneas 28 a 31).

En el `manifest.json` ponemos los datos de conexión al AP, pero tenemos que evitar que al importar Net se lance el proceso de autoconexión. Para ello tenemos que indicar que no queremos cargar `setup/network` (líneas 10 a 12):

```
1 {
2   "include": [
3     "$(MODDABLE)/examples/manifest_base.json",
4     "$(MODDABLE)/examples/manifest_net.json"
5   ],
6   "modules": {
7     "*": [
8       "./main"
```

```
    ],
    "~": [
      "$(BUILD)/devices/esp32/setup/network"
    ]
  },
  "config": {
    "ssid": "embebidos",
    "password": "22embebidos23"
  }
}
```