Internet of Things

Laborator 3

Comunicaţii Wi-Fi
(anul universitar 2022-2023)

Universitatea Tehnică ”Gheorghe Asachi” din Iaşi
Facultatea de Automatică şi Calculatoare
Departamentul de Calculatoare

Cuprins

Listă tabele

Listă figuri

1. Scopul laboratorului

În acest laborator este studiată structura unei aplicaţii care are la bază biblioteca Wi-Fi din *esp-idf* şi sunt explorate modurile de stabilire a unei conexiuni (STA/AP). De asemenea, sunt introduse câteva structuri folosite pentru sincronizarea/comunicaţia între firele de execuţie FreeRTOS.

2. Biblioteca Wi-Fi

2.1 Privire de ansamblu

Integrarea funcţiilor de comunicaţie Wi-Fi într-o aplicaţie implică folosirea mai multor componente existente în framework-ul *esp-idf*:

- **driver Wi-Fi** - la iniţializare se crează un task care se ocupă de managementul funcţiilor de nivel scăzut şi de interacţiunile cu hardware-ul.
- **buclă de evenimente Wi-Fi** - la iniţializare se crează un task care va procesa evenimente generate de driver-ul Wi-Fi sau de stiva de comunicaţie şi va rula funcţiile de callback instalate fie de biblioteci, fie de aplicaţia propriu-zisă. Această abordare a fost aleasă pentru a evita folosirea implementărilor blocante ale apelurilor API (acolo unde este cazul) sau a mecanismenlor de comunicaţie/sincronizare între firele de execuţie.
- **blbliotecă lwIP (TCP/IP)** - folosirea ei presupune crearea unuia sau mai multor task-uri (ex. în cazul conectării la un AP se va crea un task care rulează un client DHCP) în funcție de specificul aplicației.

De la pornirea aplicaţiei, până la îndeplinirea sarcinilor uzuale de comunicație, pot fi identificate un număr de etape distincte ce necesită interacţiuni specifice cu componentele API:

- **Iniţializare** - se iniţializează memoria non-volatilă (există definit o partiţie dedicată în memoria Flash SPI) în care se vor salva configurările Wi-Fi, se iniţializează stiva lwIP (TCP/IP), se crează bucla de evenimente, se iniţializează driver-ul Wi-Fi.
- **Configurare** - driver-ul Wi-Fi poate fi configurat să funcţioneze într-unul din următoarele moduri:

Tabel 1. Configuraţii Wi-Fi

Mod	Descriere
WIFI_MODE_STA	Driver-ul expune o interfaţă de tip STA (*Station*) prin intermediul căreia este posibilă conectarea la un AP (*Access Point*)
WIFI_MODE_AP	În acest mod (*softAP*) driver-ul expune o interfaţă AP şi permite conectarea unui număr configurabil de clienţi (STA). Stiva lwIP pune la dispoziţie şi un server DHCP pentru alocarea dinamică de adrese IP.
WIFI_MODE_APSTA	Mod de coexistență - Driver-ul expune două interfeţe distincte, una de tip STA, una de tip AP. Acestea pot fi folosite simultan.
WIFI_MODE_NULL	În acest mod de configurare interfaţa poate fi folosită fie pentru captura traficului, fie pentru generare de cadre raw.

- **Conectare/Aşteptare conexiuni**
  - *Mod STA* - în varianta nesupervizată, platforma iniţiază secvenţa de conectare la AP (cel puţin *SSID*-ul şi parola trebuie să fi fost anterior stocate în memoria non-volatilă). În varianta asistată, platforma scanează mediul pentru identificarea AP-urilor disponibile, utilizatorul fiind responsabil pentru selectarea reţelei şi pentru introducerea parolei
  - *Mod AP* - platforma aşteaptă cereri de conectare
- **Conexiune stabilită** - După stabilirea conexiunii (sau mai multor conexiuni, în cazul modului AP) se execută funcţiile de nivel superior specifice aplicaţiei (ex. socket-uri, client MQTT, client HTTP etc.)

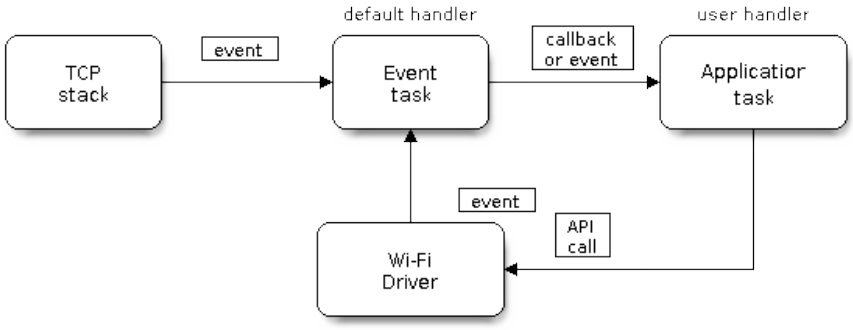

Fig. 1. Modelul de interacţiune de la nivelul unei aplicaţii Wi-Fi

2.2 Evenimente

Evenimentele generate de driver-ul Wi-Fi şi de stiva de comunicaţie sunt centralizate la nivelul buclei de evenimente (aceasta rulează pe un fir de execuţie distinct de cel pe care rulează *app_main*) şi pot fi tratate de aplicaţia finală prin intermediul funcţiilor de callback (Fig. 1). Tabelul 2 prezintă o parte dintre tipurile de evenimente generate de modulele (software) `esp_wifi`, respectiv `esp_netif`.

Tabel 2. Tipuri de evenimente (selecţie)

Eveniment	**Descriere**
ESP_EVENT_ANY_ID	Acest id se foloseşte pentru înregistrarea unei singure funcţii de callback pentru toate tipurile de evenimente generate.
WIFI_EVENT_WIFI_READY	Driver-ul Wi-Fi a încheiat iniţializarea
WIFI_EVENT_SCAN_DONE	Încheierea procesului de scanare (în mod STA)
WIFI_EVENT_STA_START	Pornirea interfeţei STA
WIFI_EVENT_STA_STOP	Oprirea interfeţei STA
WIFI_EVENT_STA_CONNECTED	Conectarea interfeţei STA la un AP
WIFI_EVENT_STA_DISCONNECTED	Deconectarea de la un AP
WIFI_EVENT_STA_AUTHMODE_CHANGE	Modificarea modului de autentificare la nivelul AP-ul la care interfaţa STA este conectată.
WIFI_EVENT_AP_START	Pornirea interfeţei AP
WIFI_EVENT_AP_STOP	Oprirea interfeţei AP
WIFI_EVENT_AP_STACONNECTED	Conectatrea unei STA la AP
WIFI_EVENT_AP_STADISCONNECTED	Deconectarea unei STA de la AP
IP_EVENT_STA_GOT_IP	Atribuirea/reatribuirea/expirarea unei adrese IPv4 (prin DHCP) sau configurarea manuală a unei adrese în modul STA.
IP_EVENT_STA_LOST_IP	Dealocarea unei adrese IP (eveniment asociat cu cel de deconectare de la un AP).
IP_EVENT_AP_STAIPASSIGNED	Alocarea unei adrese IP către o STA conectată anterior.

2.3 Modul STA

Info

Se va consulta documentaţia on-line.
Aceasta conţine o diagramă de secvenţă care prezintă interacţiunile dintre firele de execuţie ale unei aplicaţii care foloseşte interfaţa STA.
Etapele (iniţializare, configurare, conectare etc.) sunt prezentate în detaliu, accentul fiind pus pe ordinea de apel a funcţiilor API, cât şi pe tipurile de evenimente generate.

2.4 Modul AP

Info

Se va consulta documentaţia on-line.
Aceasta conţine o diagramă de secvenţă care prezintă interacţiunile dintre firele de execuţie ale unei aplicaţii care foloseşte modul AP.

2.5 Scanarea mediului

În unele aplicaţii este necesară fie aflarea tuturor SSID-urilor din raza de comunicaţie, fie descoperirea unor SSID-uri specifice (de exemplu în scenarii de provisioning).

Funcţia `esp_err_t esp_wifi_scan_start(const wifi_scan_config_t *config, bool block)` este folosită pentru iniţierea procesului de scanare. Modul de scanare este configurat prin intermediul parametrului `config`. Funcţia de scanare poate fi completă (toate canalele, toate SSID-urile, inclusiv cele ascunse) sau poate fi realizată într-o manieră limitată prin explicitarea unor parametri de configurare (ex. canale scanate, număr maxim de SSID-uri, perioadă de interogare per canal).

Notă

Pentru a forţa o scanare completă, câmpurile `ssid` şi `bssid` ale obiectului de tip `wifi_scan_config_t` se iniţializează cu pointeri nuli, iar câmpul `channel` cu valoarea 0. Alternativ, referinţa `config` poate fi nulă.

Al doilea parametru al funcţiei specifică modul de apel - blocant (`true`) sau neblocant (`false`). Pentru apelul neblocant, la sfârșitul operației se generează evenimentul WIFI_EVENT_SCAN_DONE.

Notă

Dacă sistemul este în curs de conectare la un AP (a fost apelată funcția `esp_wifi_connect`), apelul de scanare va returna o eroare.

Rezultatul scanării este returnat prin intermediul funcției `esp_err_t esp_wifi_scan_get_ap_records(uint16_t *number, wifi_ap_record_t *ap_records)`. Spaţiul pentru `ap_records` trebuie prealocat, iar parametrul `number` are dublu rol: ca parametru de intrare specifică dimensiunea spaţiului alocat pentru `ap_records`, iar ca parametru de ieşire specifică numărul de rezultate returnate.

3. Sarcini de lucru

1. Configurarea ESP32 în mod STA, stabilirea unei conexiuni cu un AP şi schimbul de date între sisteme.

- Creaţi un proiect nou şi completaţi fişierul *platformio.ini* conform indicaţiilor din laboratorul 1
- Înlocuiţi fişierul *main.c* cu cel disponibil pe platforma Moodle
- Încărcaţi şi rulaţi aplicaţia. Aceasta expune o interfaţă STA şi realizează o conexiune cu AP-ul din laborator. Un socket UDP este configurat să recepţioneze datagrame pe portul 10001
- Porniţi monitorul serial şi identificaţi adresa IPv4 atribuită platformei (listing 1). Fiecare platformă va primi o adresă IP unică.

```
I (1029) wifi: AP's beacon interval = 102400 us, DTIM period = 3
I (2089) esp_netif_handlers: sta ip: 192.168.89.10, mask: 255.255.255.0 ...
I (2089) wifi station: got ip:192.168.89.10
I (2089) wifi station: connected to ap SSID:lab-iot password: ...
```

Listing 1. Exemplu raportare alocare adresă IP

- Conectaţi calculatorul la reţeaua Wi-Fi în care este conectată şi platforma: SSID: `lab-iot`, parolă: `IoT-IoT-IoT`
- Descărcaţi de pe platforma Moodle script-ul Python *udp_sender.py*
- Editaţi scriptul şi la linia `PEER_IP = "192.168.89.xyz"` înlocuiţi textul `xyz` cu ultimul octet al adresei IP identificate în pasul anterior
- Rulaţi scriptul. Ce observaţi la nivelul monitorului serial?

2. Extindeţi aplicaţia de la punctul 1 - se va implementa controlul LED-ului conectat la GPIO4 prin UDP:
   - La recepţia unei datagrame de forma `GPIO4=0` LED-ul se va stinge, iar la recepţia unei datagrame de forma `GPIO4=1` LED-ul se va aprinde.
   - Se va modifica scriptul Python pentru a trimie alternativ cele două şiruri de octeţi.
3. Extindeţi aplicaţia de la punctul 2 - controlul LED-ului va fi realizat de pe o altă platforma ESP32:
   - Se alege un partener, se stabilesc de comun acord porturile folosite, se face schimb de adrese IP. Adresele IP şi porturile vor fi folosite în mod static (hard-coded) la nivelul codului sursă.
   - La apăsarea butonului conectat la GPIO2 se va trimite o datagramă către platforma partenerului. Datagrama va conţine alternativ textele indicate la punctul 2.

4. Scanarea mediului şi listarea AP-urilor disponibile. Creaţi un proiect nou, preluaţi şi **adaptaţi** codul demo disponibil aici.

5. Studiați funcția `select` (I/O multiplexing) şi folosiţi-o pentru verificarea buffer-ului de recepţie de la nivelul socket-ului.

Materiale de studiu

- Foaie de catalog şi manual ESP32
- Documentaţie API esp-idf
- Documentaţie API FreeRTOS