



Universitatea Politehnica Bucureşti
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect de diplomă

Integrare a unui sistem embedded cu Google Home / Amazon Alexa

Prezentat ca cerință parțială pentru obținerea
titlului de *Inginer*
în domeniul *Electronică, Telecomunicații și Tehnologia Informației*
programul de studii *Tehnologii și Sisteme de Telecomunicații*

Absolvent
Andra-Dumitrana RĂDULESCU

Conducător științific
Conf. dr. ing. Radu HOBINCU

Bucureşti, 2024

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Program de studiu **TST**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **RĂDULESCU A.G. Andra-Dumitrana, 443C**

1. Titlul temei: Integrare a unui sistem embedded cu Google Home / Amazon Alexa

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):

Studentul va dezvolta o aplicație software care să ruleze pe un sistem embedded dotat cu WiFi/Ethernet și transciever Bluetooth și care să se fie compatibil cu Google Home și Amazon Alexa. În funcție de comanda primită, sistemul va controla mai departe unul sau mai multe becuri sau lumini, inclusiv lumini decorative pentru bradul de Crăciun, prin interfața Bluetooth. Contribuțile studentului constau în: implementarea API-ului ce permite primirea de comenzi și transmiterea de răspunsuri către unitățile Google Home/ Alexa, identificarea protocolului de comunicație Bluetooth pentru cel puțin un bec sau lumini decorative folosind analiza de pachete Bluetooth și reimplementarea acestora în cadrul aplicației principale. Scopul final este realizarea unui gateway compatibil Google Home/ Alexa pentru dispozitive Bluetooth care nu sunt compatibile cu aceste tehnologii.

3. Discipline necesare pt. proiect:

PCLP1, PCLP2, SDA, MC

4. Data înregistrării temei: 2024-02-08 10:22:08

Conducător(i) lucrare,
Conf. dr. ing. Radu HOBINCU

Student,
RĂDULESCU A.G. Andra-Dumitrana

Director departament,
Conf. dr. ing. Șerban OBREJA

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: **dea9107795**

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul “*Integrare a unui sistem embedded cu Google Home / Amazon Alexa*”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *ingineriei electronice, telecomunicațiilor și tehnologiilor informaționale*, programul de studii *Tehnologii și Sisteme de Telecomunicații* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățămînt superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 27.06.2024

Absolvent *Andra-Dumitrana RĂDULESCU*



Cuprins

Glosar	9
Listă de figuri	11
Listă de tabele	12
Introducere și motivație	13
Motivație	13
Analiza State of the Art	13
IoT și automatizarea locuinței	13
Soluții de iluminat inteligente	14
1. Descrierea sistemului	15
1.1. Echipamente utilizate	15
1.1.1. Google Nest Mini	15
1.1.2. Amazon Echo Dot	16
1.1.3. Seeed Studio XIAO ESP32C3	17
1.1.4. Ghirlandă luminoasă	19
1.2. Tipuri de conexiuni utilizate	20
1.2.1. Wi-Fi	20
Istorie și evoluție	20
Concepțe cheie în tehnologia Wi-Fi	21
Standarde de securitate în Wi-Fi	22
1.2.2. Bluetooth Low Energy (BLE)	23
Diferențele dintre BLE și Bluetooth clasic	23
Arhitectura BLE	24
Atributele	26
Servicii și Caracteristici	27
Tipuri de mesaje în BLE	28
1.2.3. Conexiunea prin cablu USB	29
1.3. Integrarea cu SinricPro și Alexa/Google Home	30
1.3.1. Înregistrarea și configurarea dispozitivelor	30
1.3.2. Protocole de comunicare	31
1.3.3. Integrarea cu ajutorul kiturilor de dezvoltare (SDK-uri)	32
1.4. Ingineria inversă a pachetelor BLE	33
1.4.1. Capturarea pachetelor Bluetooth și extragerea fișierului HCI Snoop pe computer	33
1.4.2. Analiza și identificarea pachetelor relevante cu Wireshark	34
1.4.3. Testarea datelor obținute	37
2. Descrierea softwareului	39
2.1. Bibliotecile SinricPro și SinricProLight	39
2.2. Biblioteca WiFi.h	39
2.3. Biblioteca NimBLEDevice	40
2.4. Detalierea Codului Sursă	40

3. Obstacole întâlnite și soluționarea lor	47
3.1. Alegerea bibliotecilor folosite	47
3.2. Gestionarea resurselor	47
3.3. Redactarea documentației proiectului	48
4. Evaluarea Funcționalității Proiectului	49
5. Contribuții personale	51
Concluzii	53
Perspective de dezvoltare în viitor	53
Bibliografie	55
Anexe	59
A. Fișiere sursă	60

Glosar

AP Acess Point. 21

ATT Attribute Protocol, protocolul de atribut. 25

BLE Bluetooth Low Energy. 13, 15, 19, 47

EDR Enhanced Data Rate, este o parte optională a specificației Bluetooth care oferă o rată de date mai rapidă (viteză) și, eventual, o durată de viață îmbunătățită a bateriei.. 23

GAP Generic Access Profile, profilul generic de acces. 25, 26

GATT Generic Attribute Profile, profilul generic de atribute. 25

GPIO General Purpose Input/Output. 18

I2C Inter-Integrated Circuit. 17

IoT Internet of Things. 13, 17, 23, 39

JSON JavaScript Object Notation, este un format de schimb de date ușor de citit și scris de către oameni și ușor de analizat și generat de către mașini. Este bazat pe un subset al limbajului de programare JavaScript și este utilizat pentru a transmite date între un server și o aplicație web [1]. 31

L2CAP Logical Link Control and Adaptation Protocol, protocolul de control și adaptare a legăturii Logice. 25

LED Light Emitting Diode. 19

LL Link Layer, stratul de legatură. 25

MIMO Multiple Input/Multiple Output. 21

PHY Physical Layer, stratul fizic. 25

RGB Red, Green, Blue, un model de culoare care folosește combinații de lumină roșie, verde și albastră pentru a crea o gamă largă de culori. 19

SDK Software Development Kit este o colecție de instrumente software, biblioteci, documentație, exemple de cod, procese și ghiduri pe care dezvoltatorii le folosesc pentru a crea aplicații pentru platforme sau cadre specifice. 32

SMP Security Manager Protocol, managerul de securitate . 25, 26

SPI Serial Peripheral Interface. 17

SRAM Static Random Access Memory. 18

UART Universal Asynchronous Receiver / Transmitter. 17

WebSocket Protocol de comunicație care oferă canale de comunicație full-duplex printr-o singură conexiune TCP, permitând interacțiunea în timp real între un client și un server cu o latență redusă [2]. 31, 32

Wi-Fi O tehnologie care permite dispozitivelor electronice să se conecteze la o rețea locală fără fir (WLAN), folosind în general frecvențele radio de 2,4 GHz sau 5 GHz. 13, 14, 15, 16, 17, 20, 47

wireless fara fir. 14, 18, 21

Listă de figuri

1.1.	Schema de interconectare a sistemului	15
1.2.	Google Nest Mini [7]	16
1.3.	Amazon Echo Dot [7]	16
1.4.	XIAO ESP32C3 [7]	17
1.5.	Localizarea și denumirea diferitelor componente ale plăcii de dezvoltare	17
1.6.	Harta pinilor [7]	18
1.7.	Componente XIAO ESP32C3	18
1.8.	Ghirlanda luminoasă	19
1.9.	Logo Wi-Fi [11]	20
1.10.	Diagrama evoluției Wi-Fi din punct de vedere al funcționalităților suportate [13]	20
1.11.	Logo Bluetooth [22]	23
1.12.	Evoluția versiunilor Bluetooth de-a lungul timpului [24]	24
1.13.	Stiva de protocole BLE [27]	25
1.14.	Structura unui atribut [29]	27
1.15.	Logo SinricPro [31]	30
1.16.	Inregistrarea dispozitivului pe platforma SinricPro	31
1.17.	Dispozitivul înregistrat de mine afișat în interfața grafică SinricPro	31
1.18.	Developer mode	33
1.19.	Comenzile folosite pentru extragerea fișierului. - adb device afiseaza o lista a dispozitivelor conectate; - adb pull/sdcard/btsnoop_hci.log copiază fisierul pe computer	34
1.20.	Wireshark logo [35]	34
1.21.	Imaginea include pachetele de tip Write Command, evidențiate cu verde deschis. În colțul superior, dreapta, se poate observa că filtrului "btatt" a fost aplicat.	35
1.22.	Structura câmpului Value în funcție de tipul comenzi	35
1.23.	Pachetele corespunzătoare cu comenzile on, off, schimbare în alb și schimbare în culorile roșu, verde și albastru	36
1.24.	nRF Connect logo [36]	37
1.25.	Identificare UUID	37
1.26.	Exemple de testare cu ajutorul aplicației nRF Connect	38
2.1.	Logo SinricPro [31]	39
3.1.	Overleaf logo [40]	48

Listă de tabele

1.1. Corespondența între comenzi și Service UUID, Characteristic UUID și Value. . . 35

Introducere și motivație

Motivație

Principalele motive pentru care am ales acest proiect sunt interesul personal în domeniul IoT (Internet of Things) și automatizarea locuinței, relevanța și actualitatea temei, provocarea tehnică și oportunitatea de învățare.

Consider că tema aleasă este una de actualitate, întrucât automatizarea locuinței și integrarea dispozitivelor inteligente cu platforme precum Alexa și Google Home reprezintă o tendință majoră în tehnologie. Scopul principal este crearea unui modul care să comunice cu platformele populare de asistență vocală prin Wi-Fi și să controleze prin BLE dispozitive inteligente care nu beneficiază de această funcționalitate.

Analiza State of the Art

Termenul „state of the art” face referire la cel mai recent stadiu de dezvoltare al unui produs. Scopul acestei analize este de a oferi un context și o înțelegere mai profundă a tehnologiilor existente și a soluțiilor disponibile pe piață, evidențiind totodată provocările și limitările cu care se confruntă aceste soluții. Astfel, în această secțiune, vom analiza stadiul actual al cercetării și dezvoltării în domeniul automatizării locuințelor inteligente, al tehnologiilor BLE și Wi-Fi, precum și al integrării acestor tehnologii cu asistenții vocali, cum ar fi Amazon Alexa și Google Home.

IoT și automatizarea locuinței

IoT face referire la o rețea de dispozitive interconectate și unic identificabile care comunică între ele prin intermediul mai multor platforme. Termenul a fost inventat în 1999 de către Kevin Ashton, director executiv al Centrului Auto-ID din Massachusetts Institute of Technology (MIT). Acest concept a fost recunoscut oficial de UE în 2008, an în care a avut loc și prima conferință europeană IoT [3].

Conform [4], IoT reprezintă „interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications.”

Automatizarea locuinței implică utilizarea unor dispozitive concepute pentru a fi utilizate în această rețea. Acestea sunt conectate prin diverse protocoale de comunicare, iar una dintre caracteristicile esențiale ale acestora este posibilitatea de a fi controlate de la distanță, utilizând telefoane, tablete sau comenzi vocale cu ajutorul boxelor inteligente.

Utilizarea asistenților vocali precum Amazon Alexa și Google Assistant a revoluționat automatizarea locuințelor, deoarece permite interacțiunea eficientă cu dispozitivele utilizând limbajul natural.

Soluții de iluminat inteligente

În prezent, există mai multe soluții de iluminat inteligente ce pot fi controlate cu Alexa sau Google Assistant. Câteva dintre cele mai cunoscute sunt Philips Hue, LIFX și TP-Link Kasa Smart.

- Becurile inteligente Philips Hue utilizează ZigBee pentru a se conecta la un hub (Philips Hue Bridge) [5]. ZigBee este un standard bine stabilit pentru crearea de rețele wireless fiabile și cu consum redus de energie. El este conceput pentru rețele personale cu dispozitive mici și de putere redusă, cum ar fi cele utilizate în automatizarea locuințelor, control industrial și alte aplicații cu rate de date mici care necesită o durată lungă de viață a bateriei și rețele sigure [6]. În ceea ce privește posibilitatea controlului luminilor cu ajutorul asistenților vocali, hub-ul se conectează la rețea Wi-Fi prin intermediul căreia comunică cu Google Home sau Alexa, permitând această funcționalitate [5].
- Becurile inteligente LIFX se conectează direct la rețea Wi-Fi (nu necesită ajutorul unui hub ca Philips Hue). Acestea sunt direct integrate cu Alexa și Google Home prin aplicația LIFX. Dispozitivele sunt descoperite pe aceeași rețea Wi-Fi, permitând astfel transmiterea directă a comenzi către becuri.

1. Descrierea sistemului

Scopul proiectului este realizarea unui gateway compatibil cu Google Home/Alexa pentru dispozitive BLE care nu sunt compatibile cu aceste tehnologii.

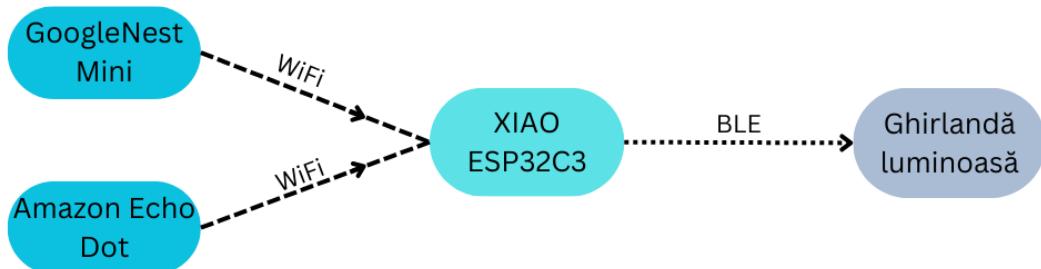


Figura 1.1.: Schema de interconectare a sistemului

1.1. Echipamente utilizate

Placa de dezvoltare primește comenzi vocale prin Wi-Fi cu ajutorul boxelor inteligente și le trimit mai departe către o ghirlandă luminoasă prin intermediul BLE. Comenzile posibile sunt:

- pornire
- oprire
- schimbare culoare

Din punct de vedere hardware, am utilizat două boxe inteligente, Google Nest Mini și Amazon Echo Dot, pentru a demonstra integrarea cu Google Home și Amazon Alexa, placă de dezvoltare Seeed Studio XIAO ESP32C3 special concepută pentru dezvoltarea în IoT și o ghirlandă luminoasă ce poate fi controlată prin BLE.

1.1.1. Google Nest Mini

Google Nest Mini, Figura 1.2, este un speaker intelligent compact, activat vocal, dezvoltat de Google. Se poate conecta la Wi-Fi și Bluetooth (versiunea 5.0), iar din punct de vedere audio, are o putere de 15W. Boxa are încorporat Google Assistant, care permite utilizatorilor să redea muzică, să gestioneze sarcinile zilnice, să obțină informații și, nu în ultimul rând, să controleze dispozitivele smart home folosind comenzi vocale. Deși Google Assistant nu oferă suport pentru limba română, comenzile pot fi transmise în limba engleză.

Conectivitatea acestuia la rețea Wi-Fi asigură comunicarea cu Seeed Studio XIAO ESP32C3, astfel făcând posibilă transmiterea comenziilor vocale către ghirlanda luminoasă.



Figura 1.2.: Google Nest Mini [7]

1.1.2. Amazon Echo Dot

Amazon Echo Dot, Figura 1.3, este un difuzor inteligent compact cu suport pentru asistentul vocal Alexa. Dispozitivul are un design simplu și elegant, cu butoane fizice pentru controlul volumului și al acțiunii, îmbunătățind interacțiunea utilizatorului. Dispune de conectivitate la Wi-Fi și Bluetooth și are funcționalități similare cu cele oferite de Google Nest Mini. Boxa dispune și de doi conectori jack de 3,5 mm și 6,3 mm, se alimentează de la rețea, iar din punct de vedere audio, are o putere de 1,2W.

În contextul acestui proiect, Echo Dot este utilizat pentru a transmite comenzi către XIAO ESP32C3, care gestionează comportamentul luminilor.



Figura 1.3.: Amazon Echo Dot [7]

1.1.3. Seeed Studio XIAO ESP32C3

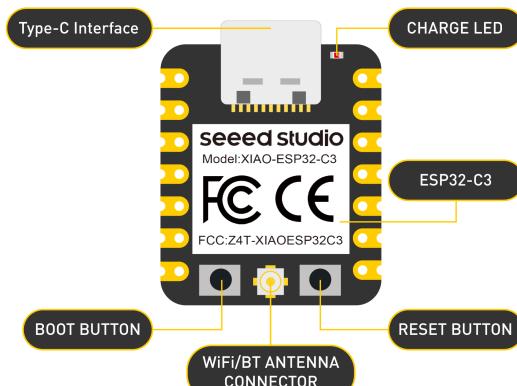
Seeed XIAO ESP32-C3 este o placă de dezvoltare versatilă, proiectată pentru aplicații IoT (Internet of Things). Aceasta combină puterea microcontrolerului ESP32-C3 [8] cu forma compactă și bogată în funcții a factorului de formă XIAO, așa cum se poate vedea în Figura 1.4. Placa are dimensiuni mici, de 20x17,5 mm, și include mai multe interfețe pentru transmisia serială a datelor, printre care UART, I2C și SPI [7].



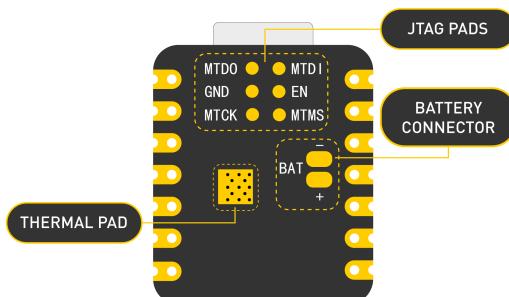
Figura 1.4.: XIAO ESP32C3 [7]

La bază, microcontrolerul ESP32-C3 [8] oferă capacitați avansate de procesare (procesor single-core RISC-V pe 32 de biți care funcționează la frecvențe până la 160 MHz), conectivitatea Wi-Fi și Bluetooth 5 (LE), făcându-l ideal pentru o gamă largă de proiecte IoT.

Următoarele imagini prezintă descrierea generală a elementelor componente ale plăcii de dezvoltare și poziția lor.



(a) Fața plăcii [9]



(b) Spatele plăcii [9]

Figura 1.5.: Localizarea și denumirea diferitelor componente ale plăcii de dezvoltare

Placa dispune de 14 pini, dintre care 11 sunt GPIO (General Purpose Input/Output), 2 pini de alimentare (5V și 3.3V) și unul GND (Ground). Cei 11 pini GPIO ai XIAO ESP32C3 sunt multiplexați și pot fi configurați conform schemei din Figura 1.6.

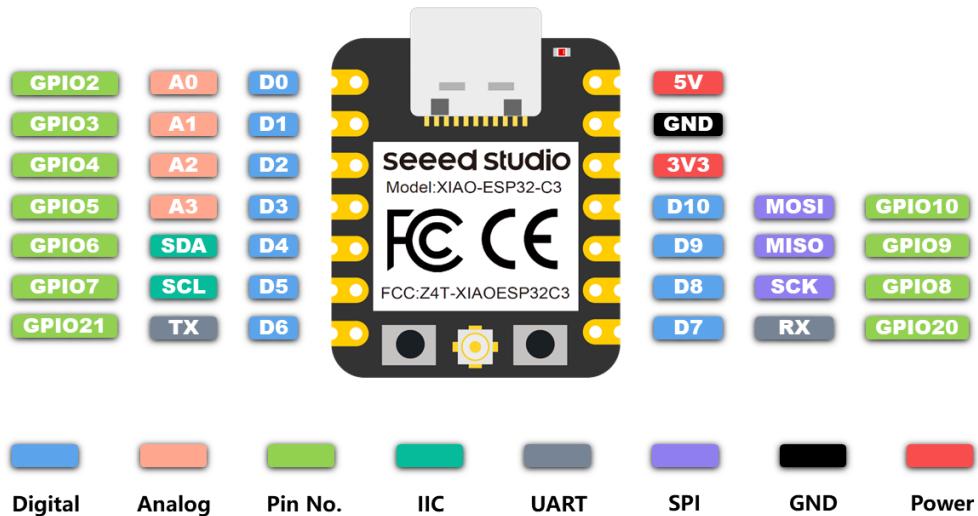


Figura 1.6.: Harta pinilor [7]

Datorită consumului redus de energie și conectivității wireless robuste, este potrivită pentru aplicații care necesită monitorizare și control la distanță [7]. De asemenea, poate fi folosită și în dispozitive alimentate cu baterie, având pe spate conectori în acest scop, cum este prezentat în Figura 1.5b. Cele două butoane, stânga (boot) și dreapta (reset), sunt ușor accesibile, iar LED-ul de culoare roșie este utilizat pentru a indica conectarea plăcii la alimentare, așa cum se poate observa în Figura 1.5a.

Din punct de vedere al memoriei, placa oferă 4MB de memorie flash integrată și 400KB de SRAM. De asemenea, dispune de un port USB Type-C (Figura 1.7b) pentru alimentare și programare. Prezența acestui port simplifică procesul de dezvoltare și permite comunicarea ușoară cu un computer pentru depanare și încărcarea firmware-ului. Placa este echipată și cu o antenă (Figura 1.7a) externă pentru a îmbunătăți puterea semnalului, conform [7].

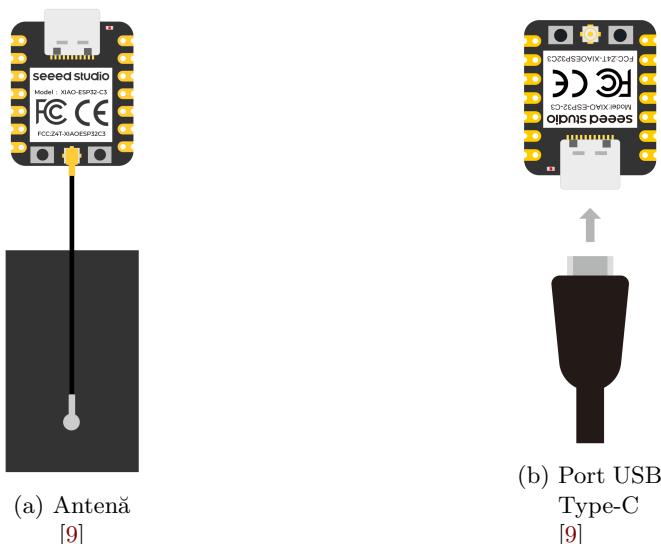


Figura 1.7.: Componente XIAO ESP32C3

1.1.4. Ghirlandă luminoasă

Ghirlanda luminoasă, Figura 1.8 folosită este compusă din 150 LED-uri RGB, un modul de control (server BLE), și poate fi controlată de diverse dispozitive folosind aplicația Actuel RGB Light [10] prin interfața lor Bluetooth. De fiecare dată când ghirlanda este conectată la alimentare, ea afișează un program de lumini prestativ (becurile își schimbă culoarea treptat, creând un efect de gradient, iar tranzițiile de culoare sunt asincrone între becuri). Atunci când ghirlanda este stinsă, ea reține setarea de culoare anterioară și când este reaprinsă revine la ea.



Figura 1.8.: Ghirlandă luminoasă

Specificații:

- **Număr Model:** RB-150-26V6W-c
- **Număr Total de LED-uri:** 150 LED-uri
- **Tensiune de Funcționare:** 24V DC
- **Consum Maxim de Energie:** 6W
- **Specificații individuale ale LED-urilor :**
 - LED-uri Roșii/Verzi/Galbene: 1.8-2.2V, 0.036-0.044W
 - LED-uri Albastre: 2.8-3.4V, 0.056-0.072W
- **Consum Total de Energie:** 8.57W (220-240V AC, 50/60Hz)

Specificațiile sursei de alimentare:

- **Număr Model:** LRLX-IP-24006
- **Intrare (PRI):** 220-240V AC, 50/60Hz
- **Ieșire (SEC):**
 - Tensiune Nominală: 24V DC
 - Putere Nominală: 6W
 - Tensiune Maximă de Ieșire: 26V DC
- **Temperatură Maximă a Carcasei (tc):** 65°C
- **Clasa de Protecție la Pătrundere:** IP44 (Protejat împotriva obiectelor solide de peste 1mm și împotriva stropirii cu apă din orice direcție)

1.2. Tipuri de conexiuni utilizate

1.2.1. Wi-Fi

Wireless Fidelity (Wi-Fi) este o tehnologie de rețea fără fir care permite dispozitivelor să comunice între ele și să acceseze internetul prin intermediul unui punct de acces sau router. Wi-Fi este un standard de rețea bazat pe specificațiile IEEE 802.11 și este utilizat pe scară largă pentru a conecta dispozitivele la rețelele locale și la internet.



Figura 1.9.: Logo Wi-Fi [11]

"The IEEE 802.11 standard for wireless local area networking (WLAN), commercially known as Wi-Fi, has become a necessity in our day-to-day life. Over a billion Wi-Fi access points connect close to hundred billion of IoT devices, smart phones, tablets, laptops, desktops, smart TVs, video cameras, monitors, printers, and other consumer devices to the Internet to enable millions of applications to reach everyone, everywhere." [12]

Istorie și evoluție

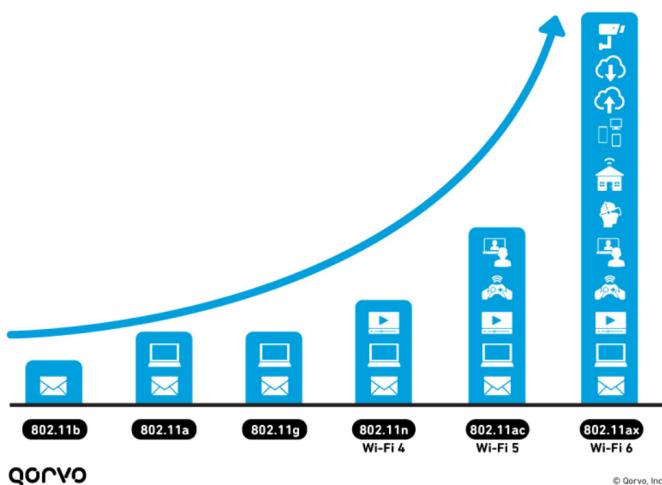


Figura 1.10.: Diagrama evoluției Wi-Fi din punct de vedere al funcționalităților suportate [13]

Wi-Fi a fost dezvoltat pe baza standardului IEEE 802.11, primul dintre acestea fiind lansat în 1997. De atunci, mai multe versiuni și îmbunătățiri ale standardului au fost introduse [13]:

- IEEE 802.11b (1999): Prima versiune larg adoptată, operând la 2.4 GHz și oferind viteze de până la 11 Mbps.
- IEEE 802.11a (1999): Operează la 5 GHz și oferă viteze de până la 54 Mbps.

- IEEE 802.11g (2003): Combină avantajele 802.11a și 802.11b, operând la 2.4 GHz cu viteze de până la 54 Mbps.
- IEEE 802.11n (Wi-Fi 4, 2009): Introduce tehnologia MIMO (Multiple Input, Multiple Output), operând la 2.4 GHz și 5 GHz și oferind viteze de până la 600 Mbps.
- IEEE 802.11ac (Wi-Fi 5, 2013): Operează doar la 5 GHz și oferă viteze de până la câțiva Gbps.
- IEEE 802.11ax (Wi-Fi 6, 2019): Operează la 2.4 GHz și 5 GHz, îmbunătățind eficiența și capacitatea rețelelor.
- IEEE 802.11be (Wi-Fi 7, 2024): Este cea mai recentă versiune de Wi-Fi și operează la 2.4 GHz, 5 GHz, și 6 GHz [14].

Concepte cheie în tehnologia Wi-Fi

Punctul de acces (AP) este un dispozitiv care creează o rețea Wi-Fi și permite altor dispozitive să se conecteze la aceasta. De obicei, este conectat la o rețea cablată și transmite un semnal de tip broadcast la care clienții se pot conecta folosind standarde wireless compatibile.

Router-ul este un element esențial în infrastructura de rețea, atât pentru rețelele locale (LAN - Local Area Network) cât și pentru rețelele largi (WAN - Wide Area Network). Rolul principal al unui router este de a direcționa pachetele de date între diferite rețele, funcționând ca un „dirijor” care se asigură că datele ajung la destinația corectă. Acesta îndeplinește funcția critică de a determina cea mai bună cale pentru ca datele să circule de la sursă la destinație, folosind tabele de rutare și protocoale de rutare pentru a determina cel mai eficient traseu pentru date [15]. Termenul **client** face referire la orice dispozitiv care se conectează la AP sau router pentru a accesa resursele de rețea sau internetul. Printre dispozitivele uzuale de tip client se numără: telefon, laptop, boxe, televizoare și imprimante.

Service Set Identifier (SSID) este un identificator unic alfanumeric atribuit fiecărei rețele Wi-Fi și reprezintă numele rețelei. Acesta permite utilizatorilor și dispozitivelor să identifice și să se conecteze la rețea corectă, mai ales în zonele unde sunt disponibile mai multe rețele Wi-Fi. El poate avea până la 32 de caractere, poate fi personalizat și, pentru a spori securitatea, poate chiar fi ascuns [16].

Multiple Input, Multiple Output (MIMO) este o tehnologie utilizată în sistemele de comunicații wireless pentru a îmbunătăți throughput-ul (capacitatea de transfer) și fiabilitatea transferului de date. Această tehnologie utilizează mai multe antene pentru a transmite și recepționa date simultan pe același canal radio. Prin exploatarea propagării multipath, MIMO permite diversitatea spațială, îmbunătățind robustețea semnalului (capacitatea semnalului radio de a rămâne stabil) și reducând efectele fading-ului și interferențelor [17].

MIMO crește semnificativ capacitatea și eficiența rețelelor wireless, permitând rate de transfer mai mari și performanțe îmbunătățite, în special în medii cu condiții radio dificile.

Fading-ul este fenomenul în care semnalul radio suferă variații în intensitate (amplitudine) sau fază pe măsură ce călătoresc de la transmitator la receptor, iar noțiunea de interferențe se referă la fenomenele sau semnalele nedeterminate care pot afecta sau distorsiona semnalul dorit.

În concluzie, putem spune că MIMO a devenit o caracteristică esențială în standardele moderne de Wi-Fi, contribuind la capacitatea acestora de a suporta simultan mai mulți utilizatori și dispozitive cu throughput crescut [17].

Standarde de securitate în Wi-Fi

Securitatea Wi-Fi este esențială pentru protejarea datelor transmise prin rețelele wireless și implică utilizarea diverselor protocole și măsuri împotriva accesului neautorizat și amenințărilor cibernetice.

Wired Equivalent Privacy (WEP) a apărut în 1997, fiind unul dintre primele protocole de securitate pentru rețelele wireless. El a fost conceput pentru a oferi un nivel de securitate comparabil cu rețelele cablate. Din punct de vedere al criptării pachetelor de date, el utilizează algoritmul RC4, dar include și un checksum (CRC-32) pentru verificarea erorilor. În ciuda promisiunilor inițiale, WEP are vulnerabilități semnificative datorită utilizării cheilor statice (implică utilizarea aceleiași chei pentru toate pachetele de date) și a verificărilor de integritate slabe, factori ce îl fac susceptibil la diverse atacuri, cum ar fi interceptarea și modificarea pachetelor și recuperarea cheilor. În prezent, instrumentele pentru spargerea criptării WEP sunt larg disponibile, ceea ce îl face o alegere nesigură și care nu ar trebui utilizată în rețelele moderne [18], [19], [20].

Wi-Fi Protected Access (WPA) a fost introdus în 2003, cu scopul de a remedia problemele de securitate ale WEP și pentru a oferi o protecție mai bună pentru rețelele Wi-Fi. Pentru criptare, el utilizează Temporal Key Integrity Protocol (TKIP), care este conceput să schimbe dinamic cheile pentru fiecare pachet transmis. Pentru a preveni manipularea pachetelor, a fost introdus mecanismul Message Integrity Check (MIC) prin care este verificată integritatea mesajelor. Deși WPA a venit cu îmbunătățiri față de WEP, acesta încă are vulnerabilități, în special legate de protocolul TKIP [18], [19], [20].

Wi-Fi Protected Access II (WPA2) a apărut în 2004 și, în prezent, este cel mai utilizat protocol de securitate Wi-Fi. El se bazează pe standardul IEEE 802.11i și oferă o securitate mai puternică decât WPA prin utilizarea criptării cu AES (Advanced Encryption Standard), un standard de criptare simetrică adoptat de guvernul SUA pentru protejarea informațiilor clasificate. De asemenea, WPA2 a introdus protocolul CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol) în locul protocolului TKIP, utilizat în WPA [20].

Wi-Fi Protected Access III (WPA3) a fost introdus în 2018 și reprezintă cea mai recentă și avansată generație de protocole de securitate Wi-Fi, concepută pentru a răspunde cerințelor de securitate moderne și pentru a oferi o protecție mai robustă decât predecesorii săi. WPA3 folosește SAE (Simultaneous Authentication of Equals), un protocol de autentificare prin care dispozitivele negociază o cheie de criptare în mod securizat, eliminând necesitatea utilizării aceleiași parole partajate pentru autentificare. Totodată, el folosește și forward secrecy, mecanism care asigură că, dacă o cheie de sesiune este compromisă, aceasta nu poate fi utilizată pentru a decripta sesiuni anterioare [19], [20].

Protected Management Frames (PMF) reprezintă o funcționalitate critică pentru securitatea rețelelor Wi-Fi, având rolul de a proteja cadrele de management împotriva interceptării și manipulării. Aceste cadre de management includ mesaje esențiale pentru funcționarea corectă a rețelei, cum ar fi autentificarea, asocierea și disocierea dispozitivelor. PMF a fost introdus ca optional în WPA2, iar în WPA3, el este obligatoriu, astfel asigurând că toate rețelele moderne beneficiază de această protecție esențială [21].

1.2.2. Bluetooth Low Energy (BLE)

Bluetooth este un standard de comunicație fără fir dezvoltat pentru comunicații pe distanțe scurte între dispozitive electronice. Tehnologia a fost numită după regele Harald "Bluetooth" Gormsson. Simbolistica din spatele acestei alegeri se bazează pe reușita regelui în a unifica Danemarca și Norvegia și astfel prin asociere tehnologia Bluetooth făcând posibilă unificarea dispozitivelor prin comunicații wireless [22].



Figura 1.11.: Logo Bluetooth [22]

Tehnologia Bluetooth a fost conceptualizată la mijlocul anilor 1990 de ingineri de la Ericsson, cu scopul de a crea o alternativă wireless la cablurile de date RS-232. În 1998, a fost înființat Bluetooth Special Interest Group (SIG), compus din companii majore precum Ericsson, IBM, Intel, Nokia și Toshiba, pentru a standardiza și promova tehnologia. Primul dispozitiv Bluetooth pentru consumatori a fost lansat în 1999, marcând începutul adoptiei sale pe scară largă [23], [24].

Primele versiuni ale Bluetooth s-au confruntat cu probleme de interoperabilitate, dar versiunile ulterioare apărute au adus îmbunătățiri semnificative.

Bluetooth 2.0 și EDR, lansat în 2004, a crescut viteza de transfer a datelor până la 3 Mbps și a îmbunătățit eficiența energetică, lucru care a determinat utilizarea acestei tehnologii în telefoanele mobile și dispozitivele audio. În 2007 a apărut Bluetooth 2.1, care a adus implementarea funcționalității Secure Simple Pairing (SSP), care a îmbunătățit procesul de conectare făcându-l simplu și mai sigur [24].

Introducerea Bluetooth 3.0 și HS (High-Speed) în 2009 a permis transferuri de date mai rapide, până la 24 Mbps (de peste 11 ori mai rapid decât versiunile anterioare). De asemenea, odată cu această versiune a fost introdus UCD (Unicast Connectionless Data), funcționalitate care a făcut dispozitivele să fie mai receptive [24], [23].

Un punct de referință major în acest domeniu a fost lansarea Bluetooth 4.0 sau Bluetooth Smart în 2010, care a introdus funcționalitatea Bluetooth Low Energy (BLE). Bluetooth Low Energy a fost conceput pentru aplicații care necesită costuri și consum de energie reduse [24], [23].

Bluetooth 5.0, introdus în 2016, a îmbunătățit și mai mult tehnologia, cu o rază de acțiune crescută, viteze de date mai mari și capacitați de difuzare îmbunătățite, făcând-o ideală pentru piața IoT [24].

În urma apariției sale în 2010, optimizarea standardului Bluetooth LE a continuat odată cu lansarea fiecărei versiuni noi. În prezent, aplicațiile care îl folosesc au un consum de energie minimal și pot funcționa pe baterii mici pentru perioade lungi de timp.

Diferențele dintre BLE și Bluetooth clasic

- **Consumul de energie:** Una dintre principalele diferențe între BLE și Bluetooth clasic este consumul de energie. După cum bine știm, BLE este conceput să funcționeze cu un

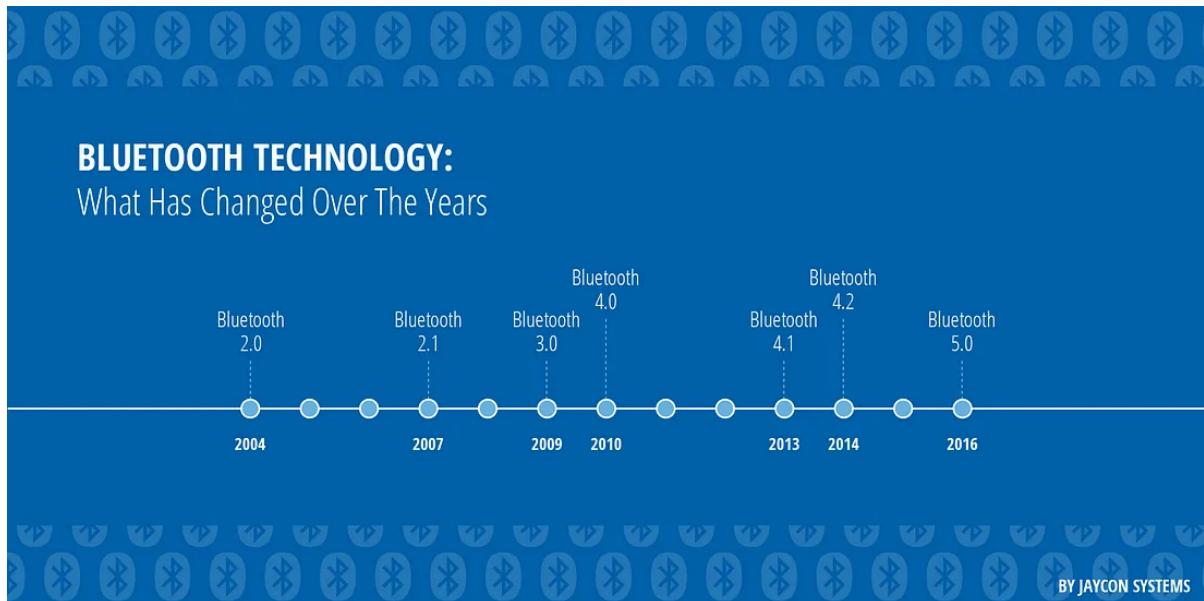


Figura 1.12.: Evoluția versiunilor Bluetooth de-a lungul timpului [24]

consum minim de energie, permitând dispozitivelor să funcționeze până la ani de zile pe o baterie mică. În schimb, Bluetooth-ul clasic este potrivit pentru aplicații care necesită schimb continuu de date, cum ar fi streaming-ul audio și transferurile de fișiere, însă are ca și consecință un consum mult mai mare de energie [25], [26].

- **Rata de transfer și raza de acțiune:** Comparativ cu Bluetooth LE, Bluetooth-ul clasic suportă rate de transfer mai mari (până la 2,1 Mbps) și este utilizat pentru aplicații care necesită transfer de date de mare viteză. În schimb, deși BLE are o rată de transfer mai mică (până la 1 Mbps în versiunile cele mai recente), excelează în scenarii unde eficiența energetică este mai importantă decât viteza. Deși dispozitivele BLE au o rază de acțiune mai mică prin comparație, acestea continuă să fie îmbunătățite [25], [26].
- **Cazuri de utilizare:** Bluetooth-ul clasic este utilizat frecvent în dispozitive precum căști wireless, boxe și alte periferice care necesită transmisie continuă de date. BLE, însă, este adaptat pentru dispozitive care trebuie să transfere periodic cantități mici de date, cum ar fi monitoarele de ritm cardiac, ceasurile inteligente și senzorii de mediu [25], [26].
- **Protocol și compatibilitate:** BLE și Bluetooth-ul clasic utilizează protocoale diferite și nu sunt direct compatibile. Pentru a depăși această limitare, au apărut dispozitive ce folosesc chip-uri Bluetooth dual-mode, care se comută continuu între cele două în funcție de tipul de Bluetooth suportat de dispozitivul cu care comunică [25], [26].

Arhitectura BLE

Arhitectura Bluetooth Low Energy (BLE) este proiectată pentru a susține comunicațiile wireless eficiente și cu consum redus de energie pentru o gamă largă de dispozitive și aplicații. Stiva de protocoale BLE poate fi împărțită în următoarele secțiuni: Controller, Host și Application, Figura 1.13.

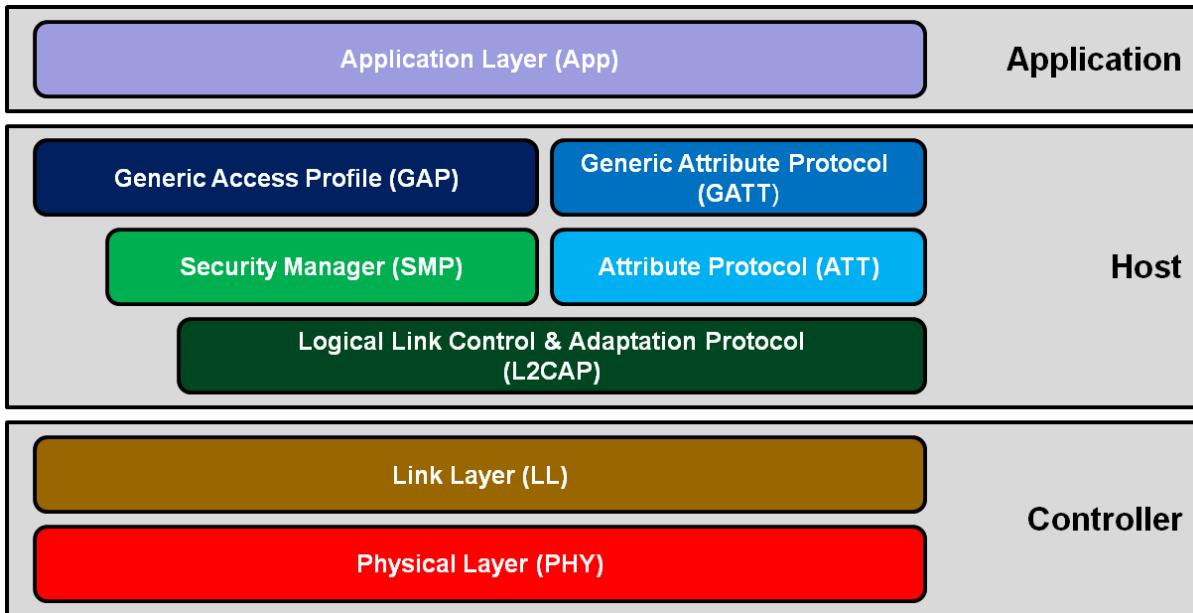


Figura 1.13.: Stiva de protocoale BLE [27]

Secțiunea Controller cuprinde Physical Layer (PHY) și Link Layer (LL).

Physical Layer (PHY) operează în banda de 2,4 GHz și gestionează sarcini precum transmisia/receptia și modularea/demodularea semnalului radio. El este esențial pentru definirea caracteristicilor fizice ale comunicării, precum canalele de frecvență și puterea semnalului. Canalele sunt în număr de 40, dintre care 37 sunt utilizate pentru transferul de date, iar 3 pentru advertising (transmiterea pachetelor de publicitate) [28], [27].

Link Layer (LL) operează direct deasupra stratului fizic (PHY) și se ocupă de sarcini precum descoperirea dispozitivelor, stabilirea și menținerea conexiunilor, precum și controlul fluxului de date. Stratul de legătură (LL) definește mai multe stări care permit dispozitivelor să se descopere reciproc, să stabilească legături și să mențină conexiunile. De asemenea, acesta gestionează sincronizarea transmisiei datelor, inclusiv gestionarea ciclurilor de somn și trezire pentru a optimiza consumul de energie și suportă mecanisme pentru comunicare securizată, inclusiv criptare și autentificare [28], [27].

Secțiunea Host gestionează protocoalele și profilurile de nivel superior, și anume Logical Link Control and Adaptation Protocol (L2CAP), Attribute Protocol (ATT), Generic Attribute Profile (GATT), Security Manager Protocol (SMP) și Generic Access Profile (GAP) [28].

L2CAP oferă multiplexarea și segmentarea datelor, asigurând că mai multe protocoale de nivel superior pot împărtăși o singură conexiune fizică. De asemenea, acesta facilitează transmiterea pachetelor de date mari prin descompunerea lor în segmente mai mici, potrivite dimensiunii pachetelor BLE, și reasamblarea lor la capătul receptor.

ATT definește modul în care datele sunt stocate și accesate pe un dispozitiv BLE. Utilizează un model client-server în care serverul deține un set de atribute, iar clientul poate citi sau scrie aceste atribute. ATT este esențial pentru organizarea datelor și asigurarea operațiunilor eficiente de citire/scriere.

GATT specifică modul în care se realizează interacțiunile între dispozitivele BLE. El este construit pe baza ATT și definește procedurile și formatele pentru descoperirea, citirea, scrierea și configurarea caracteristicilor și serviciilor unui dispozitiv. Generic Attribute Protocol este esențial pentru definirea modului în care datele sunt schimbate și asigurarea interoperabilității între dispozitive.

SMP este un protocol esențial în Bluetooth Low Energy (BLE) care se ocupă de securizarea comunicațiilor între dispozitive. El asigură o comunicare sigură gestionând procesele de autentificare, autorizare și criptare, protejând astfel datele transmise prin conexiunea BLE.

GAP definește operațiunile de bază ale unui dispozitiv BLE, cum ar fi descoperirea dispozitivelor, stabilirea conexiunilor și gestionarea conexiunilor. Tot el specifică rolurile pe care le pot avea dispozitivele și procedurile pentru conectarea și interacțiunea cu alte dispozitive BLE.

Rolurile unui dispozitiv BLE sunt:

- *Central*: Inițiază conexiuni și gestionează comunicarea cu dispozitivele periferice.
- *Peripheral*: Acceptă conexiunile inițiate de dispozitivele de tip central.
- *Broadcaster*: Trimit pachete de publicitate pentru a transmite informații, fără însă a stabili o conexiune directă.
- *Observer*: Scanează și recepționează pachetele de publicitate și, la fel ca broadcaster, nu stabilește o conexiune directă.

Secțiunea Application include Application Layer și reprezintă nivelul la care aplicațiile utilizatorului interacționează cu stiva de protocoale BLE pentru a trimite și primi date.

Această arhitectură permite BLE să mențină un consum redus de energie, oferind în același timp un set robust de funcții, făcându-l ideal pentru aplicații IoT, tehnologie purtabilă și alte dispozitive care necesită o durată prelungită de viață a bateriei și conectivitate fiabilă.

Atributele

Atributele sunt componente fundamentale în protocolul Bluetooth Low Energy (BLE), jucând un rol crucial în modul în care datele sunt organizate, transmise și accesate între dispozitive. Ele sunt definite în cadrul Attribute Protocol (ATT) și sunt structurate într-un mod care facilitează comunicarea eficientă între dispozitivele BLE.

Structura generală a unui atribut (Figura 1.14) include următoarele componente cheie [28], [29]:

- **Handle** este un identificator unic de 16 biți care permite clientului să acceseze rapid și precis attributele întă în operațiunile de citire și scriere, precum și pentru a primi notificări. Utilizarea handle-urilor simplifică și eficientizează comunicarea între client și server, asigurând că datele sunt accesate și modificate într-un mod corect și sigur.
- **Tipul atributului (Attribute Type)** este reprezentat de un UUID (Universally Unique Identifier), un identificator esențial în protocolul Bluetooth Low Energy (BLE), utilizat pentru a asigura că tipurile de attribute sunt unice și standardizate la nivel global. Scopul acestei standardizări este să împiedice existența unor dispozitive sau servicii cu același identificator pentru tipuri de attribute diferite. UUID-urile pot fi scurte (16 biți) sau extinse (128 biți), unde cele de 16 biți sunt derive din UUID-urile extinse și sunt utilizate pentru identificatorii standard definiți de Bluetooth SIG. UUID-urile de 128 biți permit definirea unor identificatori unici pentru aplicații personalizate, fără riscul de coliziuni cu UUID-urile existente, asigurând totodată că aceste attribute personalizate sunt unice.
- **Valoarea Atributului (Attribute Value)** constituie datele reale stocate într-un atribut, acestea putând reprezenta orice tip de informație, de la măsurători de senzori până la setări de configurație. Formatul (tipul de date) și lungimea valorii atributului sunt determinate

de UUID-ul atributului. Câteva dintre formatele posibile sunt siruri de caractere, numere întregi, array-uri de octeți și alte formate complexe, astfel asigurând flexibilitatea necesară pentru diverse aplicații.

- **Permisiunile Atributelor (Attribute Permissions)** specifică ce tipuri de acces sunt permise pentru fiecare atribut, asigurând astfel un acces controlat și securizat. Permisiunile pot fi împărțite în trei categorii.

- *Permisiuni de acces*: citire (permite dispozitivelor citirea valorii atributului), scriere (permite dispozitivelor scrierea unei noi valori în atribut), citire și scriere (permite atât citirea, cât și scrierea unei valori în atribut).
- *Permisiuni de autentificare*: cu autentificare, fără autentificare.
- *Permisiuni de autorizare*: cu autorizare, fără autorizare.

Permisiunile oferă un control detaliat asupra modului în care datele pot fi accesate și modificate, asigurând conformitatea cu cerințele aplicației și ale utilizatorului și prevenind modificările neintenționate sau malicioase care ar putea destabiliza funcționarea dispozitivului.

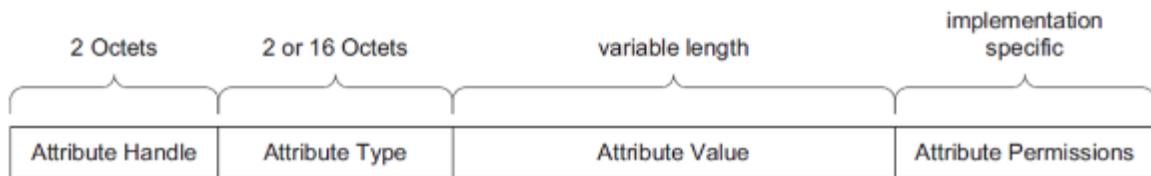


Figura 1.14.: Structura unui atribut [29]

În cadrul Profilului Generic de Atribute (GATT), atrbutele sunt grupate într-un tabel de atrbute, într-o ordine specifică și logică. Tabelul de atrbute reprezintă o structură care conține toate atrbutele utilizate de un dispozitiv BLE.

Servicii și Caracteristici

Caracteristicile reprezintă un punct unic de date sau o colecție de date asociate, permitând stocarea și gestionarea datelor într-un mod structurat și accesibil.

Elementele cheie ale unei caracteristici [28], [30]:

- **Declarația caracteristicii (Characteristic Declaration)** este un atrbut ce definește cum poate un client să interacționeze cu aceasta. Tipul acestui atrbut este UUID-ul 0x2803, utilizat doar pentru declarațiile caracteristicilor. Valoarea acestui atrbut include informații despre operațiunile permise, identificatorul unic (UUID) al caracteristicii și handle-ul atrbutului Characteristic Value. Aceste informații sunt esențiale pentru a asigura un acces corect și securizat la date.
- **Valoarea Caracteristicii (Characteristic Value)** este un atrbut ce conține datele reale transmise între dispozitivele BLE. Aceste date pot varia de la măsurători ale senzorilor (de exemplu, temperatură, ritm cardiac) la setări de configurare (de exemplu, niveluri de putere, moduri de operare) sau informații de stare (de exemplu, pornit/oprit).
- **Descriptorii (Descriptors)** oferă informații suplimentare despre caracteristici, cum ar fi unități de măsură sau limite de interval.

Serviciile în Bluetooth Low Energy (BLE) sunt colecții de caracteristici și date asociate care definesc funcționalitatea principală a unui dispozitiv BLE. Ele oferă o grupare logică a datelor și operațiunilor, facilitând gestionarea și interacțiunea cu dispozitivul.

Tipuri de mesaje în BLE

În Bluetooth Low Energy (BLE), comunicarea între dispozitive se bazează pe Protocolul de Atribut (ATT). Acest protocol definește mai multe tipuri de mesaje care facilitează interacțiunea între un client și un server BLE [28].

Cereri și Răspunsuri de Citire:

- *Read Request* este un mesaj trimis de client pentru a citi valoarea unui atribut de la server.
- *Read Response* este mesajul trimis de server ca răspuns la o cerere de citire (read request), conținând valoarea atributului solicitat.
- *Read Blob Request* este un tip de mesaj utilizat când valoarea atributului este prea mare pentru a fi inclusă într-un singur răspuns de citire (read response). În general, este folosit pentru a citi valori lungi ale unei caracteristici sau ale unor descriptori.
- *Read Blob Response* conține o porțiune din valoarea atributului și este răspunsul la o cerere de citire parțială (read blob request).

Cereri și Comenzi de Scriere:

- *Write Request* este un mesaj trimis de client pentru a scrie o valoare nouă într-un atribut de pe server. Această operațiune necesită o confirmare din partea serverului.
- *Write Response* reprezintă răspunsul trimis de server pentru a confirma o cerere de scriere (write request).
- *Write Command* este un tip de mesaj similar cu cererea de scriere, dar care nu necesită o confirmare din partea serverului. Eliminarea necesității unui mesaj de confirmare îl face mai rapid, însă are ca și consecință negativă faptul că devine mai puțin fiabil.
- *Signed Write Command* este un mesaj similar cu comanda de scriere, dar care include o semnătură criptografică pentru securitate suplimentară.

Notificări și Indicații:

- *Notification* este un tip de mesaj trimis de server pentru a informa clientul despre o schimbare în valoarea unui atribut. Notificările nu necesită o confirmare din partea clientului.
- *Indication* este un tip de mesaj cu o funcționalitate similară cu cea a notificării, dar necesită o confirmare din partea clientului. Această adăugire asigură o livrare fiabilă a mesajelor.

Cereri și Răspunsuri de Găsire a Informațiilor:

- *Find Information Request* este un mesaj trimis de client pentru a descoperi handle-urile și tipurile de atribute de pe server.
- *Find Information Response* este mesajul trimis de server ca răspuns la o cerere de găsire a informațiilor (find information request).

Cereri și Răspunsuri de Găsire după Tip și Valoare:

- *Find by Type Value Request* reprezintă mesajul utilizat de client pentru a găsi atrbute cu un anumit tip și o anumită valoare.
- *Find by Type Value Response* reprezintă mesajul trimis de server ca răspuns la cererea de găsire după tip și valoare (find by type value request), listând atrbutele care se potrivesc.

1.2.3. Conexiunea prin cablu USB

Seeed Studio XIAO ESP32C3 dispune de un port USB Type-C care facilitează comunicarea între placă și un computer. Această conexiune USB este esențială pentru alimentarea cu energie, încărcarea firmware-ului și comunicația serială cu scopul de depanare.

Conexiunea USB reprezintă sursa principală de alimentare pentru XIAO ESP32C3 în timpul dezvoltării și depanării, portul fiind capabil să furnizeze suficientă energie pentru a alimenta microcontrolerul și perifericele conectate. Aceasta elimină necesitatea unei surse de alimentare externe suplimentare, simplificând astfel procesul de dezvoltare.

Pe de altă parte, ea permite și încărcarea firmware-ului. Prezența bootloader-ului de pe ESP32-C3 facilitează acest proces, permitând plăcii să comute automat între modul bootloader și modul de operare normal atunci când detectează că un firmware este încărcat.

De asemenea, conexiunea USB oferă o interfață serială pentru comunicația între XIAO ESP32C3 și computer. Existența acestei interfețe este esențială pentru dezvoltatori, deoarece permite monitorizarea în timp real a activității microcontrolerului și depanarea eventualelor probleme. Comunicația serială este gestionată, de obicei, prin intermediul unui bridge USB-to-UART, permitând o conectivitate robustă și fiabilă.

USB to UART Bridge este o componentă hardware care facilitează transferul de date între o interfață USB și o interfață UART. Convertirea semnalelor USB în semnale UART permite comunicarea directă între computer și microcontroler prin pinii UART. Acest "pod" elimină necesitatea de a folosi un hardware suplimentar pentru convertirea semnalelor, simplificând astfel procesul de dezvoltare.

În general, procesul de încărcare a firmware-ului implică următorii pași:

1. Compilarea codului.
2. Intrarea în modul bootloader (în cazul meu, acest proces este gestionat automat de bootloader și nu necesită neapărat o acțiune din partea mea).
3. Încărcarea propriu-zisă a firmware-ului (datele din fișierul binar obținut în urma compilării sunt transmise prin USB și apoi scrise în memoria flash a microcontrolerului).

1.3. Integrarea cu SinricPro și Alexa/Google Home

SinricPro este o platformă IoT concepută pentru a facilita crearea și gestionarea dispozitivelor smart home. Ea permite dezvoltatorilor să-și integreze dispozitivele cu asistenți vocali populari precum Amazon Alexa și Google Assistant, și prin urmare posibilitatea să le controleze cu ajutorul comenziilor vocale. Pentru a utiliza această platformă este necesară crearea unui cont în care putem înregistra dispozitivele dorite [31].

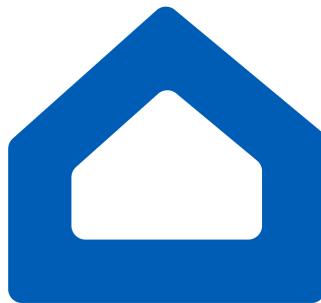


Figura 1.15.: Logo SinricPro [31]

Functionalități și caracteristici SinricPro:

- *Compatibilitatea cu mai multe platforme:* Dezvoltatorii pot scrie coduri în diferite medii de dezvoltare și pot folosi SinricPro.h pentru a integra rapid și eficient dispozitivele lor IoT.
- *Integrarea cu asistenți vocali:* SinricPro.h este optimizat pentru a funcționa cu asistenți vocali populari precum Amazon Alexa și Google Assistant.
- *Managementul Dispozitivelor:* Platforma permite înregistrarea, configurarea și usoara gestionare a dispozitivelor prin intermediul dashboard-ului SinricPro. Utilizatorii pot adăuga, elimina și configura dispozitivele conectate dintr-un singur cont SinricPro.
- *Comunicație în timp real:* SinricPro folosește WebSocket-uri pentru comunicația în timp real între dispozitiv și cloud-ul SinricPro, asigurând tempi de răspuns rapizi pentru comenzi și actualizări de stare.
- *Securitate și autentificare:* Include funcționalități de autentificare și criptare pentru a asigura securitatea comunicațiilor între dispozitive și serverul cloud.

În contextul proiectului meu, SinricPro este utilizat împreună cu placa de dezvoltare XIAO ESP32-C3 pentru a crea o poartă de acces care integrează dispozitive Bluetooth necompatibile cu Google Home și Amazon Alexa. Platforma gestionează comunicarea între asistenții vocali și placă, asigurând un control și o monitorizare fără întreruperi.

1.3.1. Înregistrarea și configurarea dispozitivelor

Adăugarea unui nou dispozitiv pe platformă și configurarea lui implică selectarea tipului de dispozitiv și numirea lui, completarea unei scurte descrieri a dispozitivului și selectarea camerei în care se află, Figura 1.16. Funcționalitățile dispozitivului se bazează în general pe tipul selectat, existând și posibilitatea definirii unui dispozitiv custom. Din punct de vedere al securității, dispozitivele primesc chei API unice și ID-uri de dispozitiv necesare pentru comunicarea securizată [31].

The screenshot shows a registration form for a device named "Fairy Lights". The form is divided into four tabs: 1. Device Information, 2. Notifications, 3. Timers, and 4. Other. The first tab is active. The fields filled in are:

- Device Name:** Fairy Lights
- Description:** RGB lights
- Device Type:** Smart Light Bulb
- App Key:** default
- Room:** Living Room (Home)

A note at the bottom of the form says: "We recommend using different App Key for different hardware modules". At the bottom right, there are links for "How to connect my device ?" and "Utterances For Alexa & Google Home". A "Next" button is visible at the bottom left.

Figura 1.16.: Inregistrarea dispozitivului pe platforma SinricPro

Dispozitivul înregistrat de mine se numește Fairy Lights și este de tip Smart Light Bulb. Printre funcționalitățile suportate de acest tip de dispozitiv, comune cu cele suportate de ghirlanda luminoasă, se numără aprinderea becului (în cazul meu ghirlanda luminoasă), stingerea becului și schimbarea culorii.

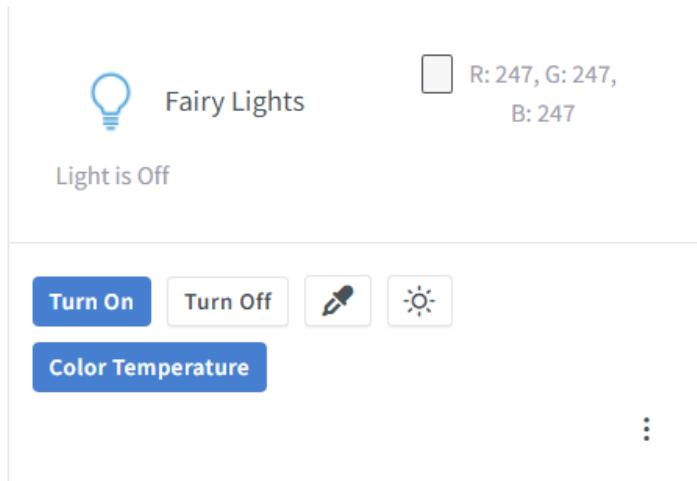


Figura 1.17.: Dispozitivul înregistrat de mine afișat în interfața grafică SinricPro

1.3.2. Protocole de comunicare

SinricPro utilizează WebSockets pentru comunicare bidirectională în timp real între dispozitivele IoT și cloud-ul SinricPro, asigurând astfel mesaje securizate și autentificate prin utilizarea cheilor HMAC. Datele schimbate sunt în format JSON, iar platforma suportă trei tipuri de mesaje: Cereri (comenzi de la platformă), Răspunsuri (confirmări de la dispozitiv) și Evenimente (notificări generate de dispozitiv) [31].

1.3.3. Integrarea cu ajutorul kiturilor de dezvoltare (SDK-uri)

SinricPro oferă kituri de dezvoltare software (SDK-uri) pentru diverse platforme, cu scopul de a simplifica procesul de integrare. Acest lucru este realizat prin gestionarea conexiunilor, analizarea mesajelor și gestionarea procesului de autentificare și permite dezvoltatorilor să se concentreze pe funcționalitățile de bază ale dispozitivelor lor. Prin conectarea SinricPro la Amazon Alexa sau Google Assistant, utilizatorii își pot controla dispozitivele folosind comenzi vocale, platforma traducând aceste comenzi în mesaje executabile trimise dispozitivelor prin conexiuni WebSocket [31].

1.4. Ingineria inversă a pachetelor BLE

Ingineria inversă este procesul de descompunere a unui produs sau sistem pentru a înțelege designul, funcționalitatea și modul său de operare. Aceasta reprezintă o practică frecvent utilizată în dezvoltarea de software, ingineria hardware și securitatea cibernetică și implică analizarea detaliată a componentelor și funcționării sistemului, adesea pentru a replica sau îmbunătăți designul original, sau pentru a înțelege cum a fost construit [32].

1.4.1. Capturarea pachetelor Bluetooth și extragerea fișierului HCI Snoop pe computer

Pentru a captura traficul Bluetooth și, respectiv, comenziile trimise de aplicația Actuel RGB Lights [10] către ghirlanda luminoasă, am utilizat opțiunile pentru dezvoltatori ale unui telefon cu Android.

Pentru a activa aceste setări, am accesat Settings > About phone > Software information > Build number, pe care am apăsat de șapte ori. Dacă activarea reușește, pe ecran va apărea mesajul "Developer mode on!" (metoda de activare și mesajul pot dифeи в функции de dispozitiv) iar în meniu Settings va apărea secțiunea Developer options (Figura 1.18a) [33].

Pentru a captura traficul, am activat opțiunea "Enable Bluetooth HCI snoop log", prezentă în Figura 1.18b, după care am trimis o serie de comenzi către ghirlanda luminoasă din interfața grafică a aplicației Actuel RGB Lights, având grija să le notez pentru a le putea identifica mai ușor.

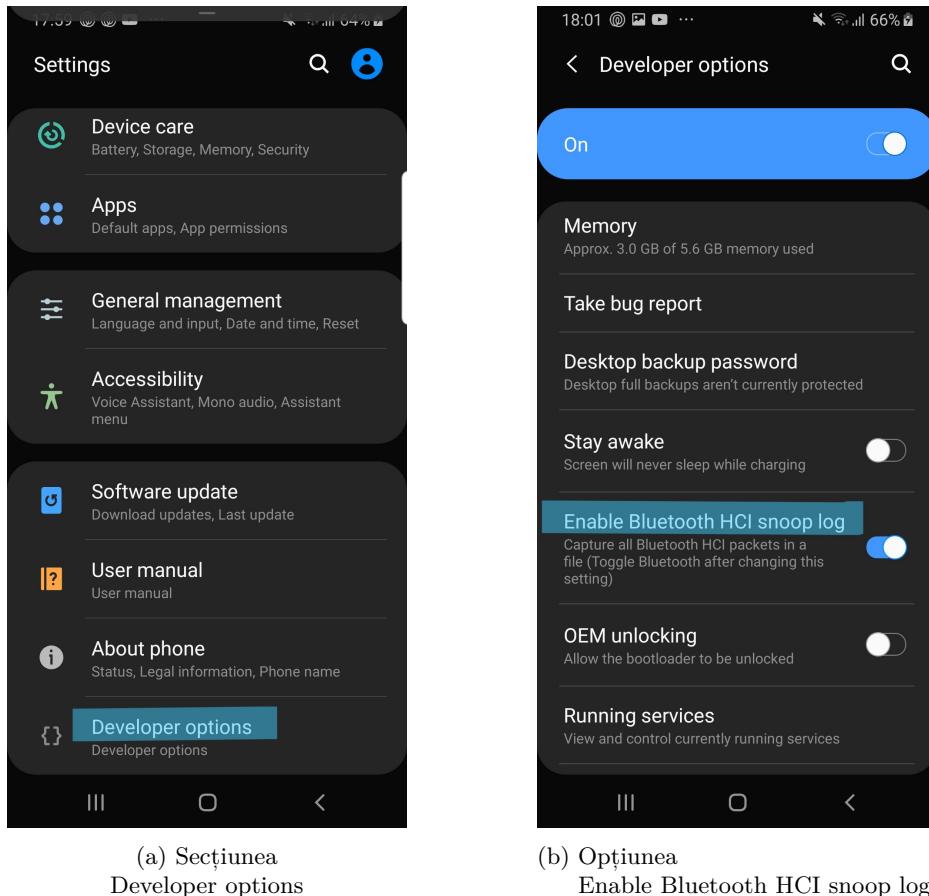
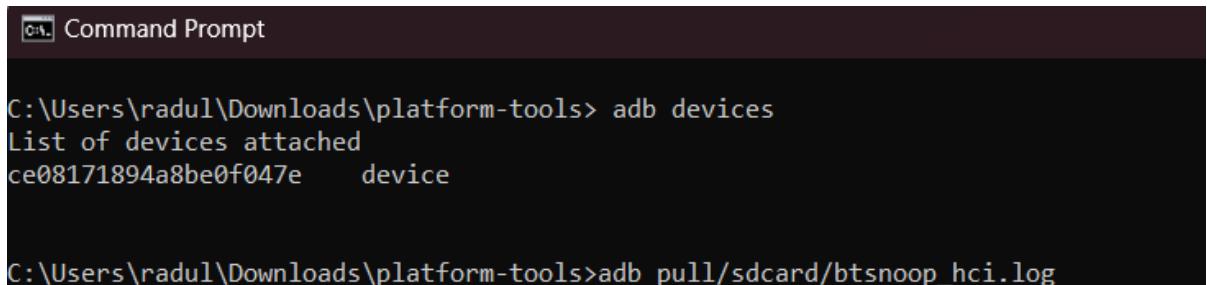


Figura 1.18.: Developer mode

Pentru a transfera jurnalul Bluetooth pe computer am folosit Android Debug Bridge (ADB), un instrument de linie de comandă care permite comunicarea cu un dispozitiv Android. El face parte din Android SDK Platform-Tools și funcționează prin USB sau rețea pentru a facilita interacțiunea directă cu dispozitivul Android [34]. Pentru ca operațiunea de transfer să aibă loc cu succes, trebuie să ne asigurăm că jurnalul a fost generat și să identificăm corect calea către acesta.



```
C:\Users\radul\Downloads\platform-tools> adb devices
List of devices attached
ce08171894a8be0f047e    device

C:\Users\radul\Downloads\platform-tools>adb pull/sdcard/btsnoop_hci.log
```

Figura 1.19.: Comenzi folosite pentru extragerea fișierului.

- adb device afiseaza o lista a dispozitelor conectate;
- adb pull/sdcard/btsnoop_hci.log copiaza fisierul pe computer

1.4.2. Analiza și identificarea pachetelor relevante cu Wireshark

Wireshark este un instrument conceput pentru analizarea protocolelor de comunicații, care permite capturarea și navigarea interactivă a traficului dintr-o rețea. Totodată, Wireshark este un instrument puternic pentru depanarea rețelelor, analiza, dezvoltarea de software, dar și în educație. Acesta are capacitatea de a inspecta detaliat sute de protocoale și poate captura pachete de date în timp real, afișând informații detaliate despre fiecare pachet. Instrumentul suportă diverse tipuri de rețele din punct de vedere al mediului de transmisie, unele dintre cele mai cunoscute fiind Ethernet, Wi-Fi și Bluetooth [35].



Figura 1.20.: Wireshark logo [35]

În contextul proiectului de diplomă, am utilizat acest program pentru a analiza fișierul btsnoop_hci.log. Pentru a identifica pachetele relevante, am aplicat filtrul "btatt", utilizat pentru a analiza pachete Bluetooth Attribute Protocol (ATT, protocol folosit de dispozitivele ce folosesc tehnologia BLE, în comunicarea dintre server și client). Am observat prezența unor pachete de tip Write Command și le-am aplicat un filtru de culoare verde deschis pentru a fi mai vizibile, așa cum se poate vedea în Figura 1.21.

Informațiile extrase din analizarea pachetelor, dar și corelația între acestea și comanda selectată sunt centralizate în Tabelul 1.1. Atât denumirile, cât și valorile din tabel au fost extrase direct din interfața Wireshark. O mare parte din valorile extrase sunt prezente și în Figura 1.23. Corespondența dintre comenzi și valori a fost făcută comparând ordinea de transmitere a comenziilor din aplicația Actuel RGB Lights cu ordinea pachetelor de tip Write Command.

btspoon_hci.log							
No.	Time	Source	Destination	Protocol	Length	Info	
424	16.256859	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	14	Sent Find Information	
426	16.353151	TexasInstrum_5e:40:... SamsungElect_13:da:d8 (GT-I9195)		ATT	15	Rcvd Find Information	
427	16.353517	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	14	Sent Find Information	
429	16.450633	TexasInstrum_5e:40:... SamsungElect_13:da:d8 (GT-I9195)		ATT	14	Rcvd Error Response	
432	16.510026	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	21	Sent Write Command	
435	18.973020	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
437	26.326751	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
439	26.738471	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
441	28.267725	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
443	52.038547	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	17	Sent Write Command	
445	53.336182	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	17	Sent Write Command	
447	53.874470	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	17	Sent Write Command	
449	54.501877	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	17	Sent Write Command	
451	56.237937	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	17	Sent Write Command	
453	64.481215	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
455	64.891744	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
457	65.302884	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
459	65.595941	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
461	65.917899	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
463	67.567404	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	
465	75.689791	SamsungElect_13:da:... TexasInstrum_5e:40:34 (Light)		ATT	15	Sent Write Command	

Figura 1.21.: Imaginea include pachetele de tip Write Command, evidențiate cu verde deschis. În colțul superior, dreapta, se poate observa că filtrului "btatt" a fost aplicat.

Comanda	Service UUID	Characteristic UUID	Value
off	0x1000	0x1001	3c0100
on	0x1000	0x1001	3c0101
alb	0x1000	0x1001	3c027f7f7f
albastru	0x1000	0x1001	3c0200007f
verde	0x1000	0x1001	3c02007f00
roșu	0x1000	0x1001	3c027f0000
mov	0x1000	0x1001	3c027f007f
albastru deschis	0x1000	0x1001	3c02007f7f
galben	0x1000	0x1001	3c027f7f00

Tabelul 1.1.: Corespondență între comenzi și Service UUID, Characteristic UUID și Value.

Pe baza coloanelor Comanda și Value putem face urmatoarele observații:

- În cazul comenzilor on, off, campul Value este alcătuit din 3 octeți, primii doi au o valoare constantă, iar ultimul variază. În concluzie, valoarea ultimului octet controlează starea dispozitivului (ghirlanda luminoasă). Atunci cand acesta are valoarea 01 comanda trimisă este "On", iar cand valoarea lui este 00, comanda trimisă este "Off".
- În cazul comenzilor de schimbare culoare și schimbare în alb, campul Value este compus din cinci octeti. Primi doi octeți au o valoare constantă, iar urmatorii trei variază și prin urmare controlează starea dispozitivului. Analizând concret corelația dintre culoarea setată și valorile ultimelor trei octeți putem concluziona că valorile stocate de ei reprezintă valorile componentelor RGB.

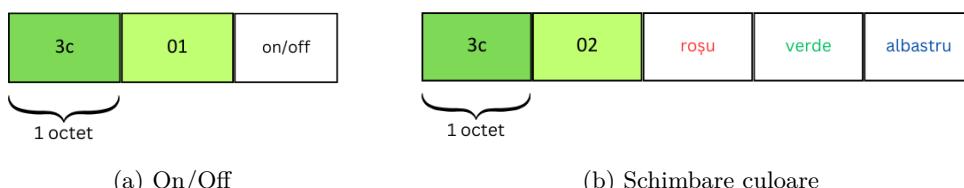


Figura 1.22.: Structura câmpului Value în funcție de tipul comenzi

```

▶ Frame 455: 15 bytes on wire (120 bits), 15 bytes captured (120 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c0101

```

(a) On

```

▶ Frame 453: 15 bytes on wire (120 bits), 15 bytes captured (120 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c0100

```

(b) Off

```

▶ Frame 443: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c027f7f7f

```

(c) Schimbare în alb

```

▶ Frame 445: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c027f0000

```

(d) Schimbare în roșu

```

▶ Frame 447: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c02007f00

```

(e) Schimbare în verde

```

▶ Frame 449: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)
▶ Bluetooth
▶ Bluetooth HCI H4
▶ Bluetooth HCI ACL Packet
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Command (0x52)
  ▶ Handle: 0x0032 (Service Discovery Server Service Class ID: Browse Group Descriptor Service Class ID)
    [Service UUID: Service Discovery Server Service Class ID (0x1000)]
    [UUID: Browse Group Descriptor Service Class ID (0x1001)]
  Value: 3c0200007f

```

(f) Schimbare în albastru

Figura 1.23.: Pachetele corespunzătoare cu comenziile on, off, schimbare în alb și schimbare în culorile roșu, verde și albastru

1.4.3. Testarea datelor obținute

Pentru a confirma corectitudinea asocierii între comanda transmisă și pachetul reprezentativ, am folosit aplicația nRF Connect for Mobile, care este special concepută pentru dezvoltarea și testarea dispozitivelor BLE [36].



Figura 1.24.: nRF Connect logo [36]

Folosind interfața grafică a aplicației, m-am conectat la ghirlanda luminoasă, după care am identificat UUID-ul serviciului și respectiv UUID-ul caracteristicii, extrase anterior (Figura 1.25).

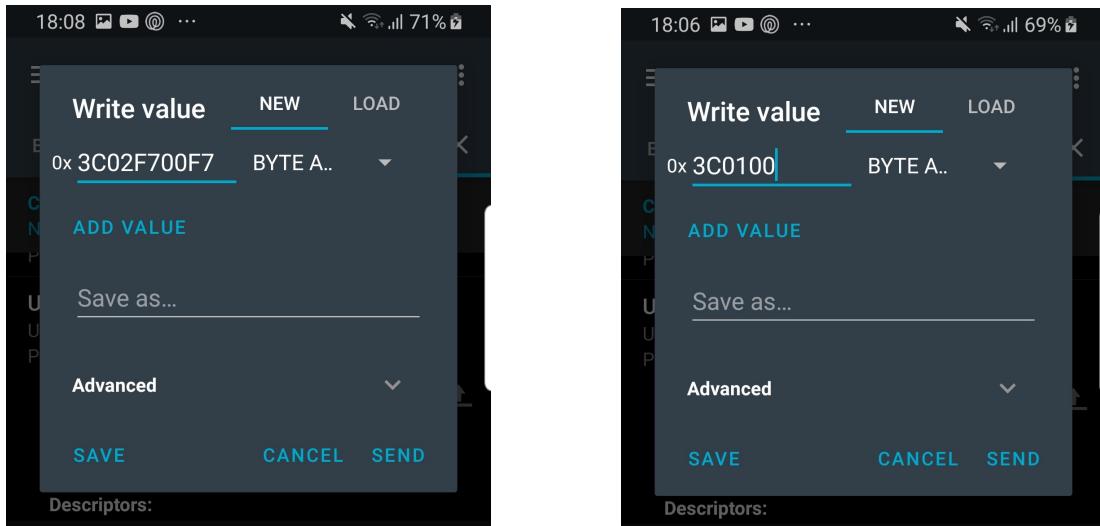
 The image shows two screenshots of the nRF Connect mobile application interface.
 The left screenshot (a) shows the 'Devices' screen with a bonded advertiser named 'LIGHT' (74:D2:85:5E:40:34). It lists several services: 'Unknown Service' (UUID: f000ff0-0451-4000-b000-000000000000), 'Unknown Service' (UUID: f000ffd0-0451-4000-b000-000000000000), 'Generic Access' (UUID: 0x1800), 'Generic Attribute' (UUID: 0x1801), 'Device Information' (UUID: 0x180A), and another 'Unknown Service' (UUID: 0x1000).
 The right screenshot (b) shows the details for the 'LIGHT' device. It lists three characteristics: 'Unknown Characteristic' (UUID: 0x1001, properties: READ, WRITE NO RESPONSE, descriptors: Characteristic User Description (UUID: 0x2901)), 'Unknown Characteristic' (UUID: 0x1002, properties: NOTIFY, descriptors: Client Characteristic Configuration (UUID: 0x2902) and Characteristic User Description (UUID: 0x2901)), and 'Unknown Characteristic' (UUID: 0x1003, properties: WRITE, descriptors: Characteristic User Description (UUID: 0x2901)).
 Both screenshots show the status bar with battery level (67% and 100%), signal strength, and time (18:03 and 19:50).

(a) Identificare UUID serviciu

(b) Identificare UUID caracteristică

Figura 1.25.: Identificare UUID

Testarea s-a realizat prin transmiterea datelor din câmpul Value al tabelului 1.1, prin intermediul interfeței grafice (Figura 1.26), sub forma unor vectori de valori hexazecimale. Astfel, am constatat că asocierile dintre servicii și comenzi au fost făcute corect.



- (a) Transmisia datelor corespunzătoare comenzi de schimbare a culoarii în mov
 (b) Transmisia datelor corespunzătoare comenzi off

Figura 1.26.: Exemple de testare cu ajutorul aplicatiei nRF Connect

În urma analizei și testării, putem spune cu certitudine că structura datelor transmise, ilustrată în Figurile 1.22a și 1.22b, precum și impactul lor asupra stărilor dispozitivului BLE, au fost identificate corect.

2. Descrierea softwareului

2.1. Bibliotecile SinricPro și SinricProLight

Biblioteca SinricPro.h este o componentă esențială a ecosistemului SinricPro. Ea oferă o interfață cuprinzătoare pentru integrarea unei game largi de dispozitive IoT (Internet of Things) cu platforma cloud SinricPro.

La fel ca biblioteca SinricPro.h, SinricProLight.h face parte din ecosistemul SinricPro, însă este special concepută pentru integrarea și controlul dispozitivelor de iluminat.

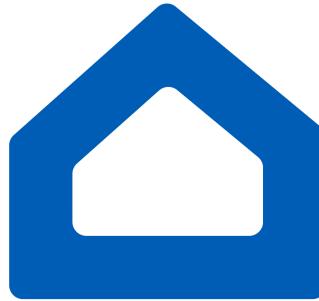


Figura 2.1.: Logo SinricPro [31]

Atât SinricPro.h cât și SinricProLight.h utilizează caracteristicile programării orientate pe obiecte ale limbajului C++ pentru a oferi un cadru modular și extensibil, permitând astfel dezvoltatorilor să gestioneze cu ușurință diferite tipuri de dispozitive de iluminat și funcționalități.

Împreună, cele două biblioteci facilitează conexiunea între dispozitivele de iluminat inteligente și asistenții vocali precum Amazon Alexa și Google Assistant, oferind astfel posibilitatea de a le controla cu ajutorul comenziilor vocale și de a le gestiona în timp real [31].

2.2. Biblioteca WiFi.h

Biblioteca WiFi.h a fost concepută pentru dezvoltarea de proiecte IoT care necesită conectivitate WiFi. Ea este utilizată pe scară largă și oferă funcții și instrumente necesare pentru gestionarea conexiunilor WiFi și interacțiunea cu rețelele.

WiFi.h oferă un set de funcții pentru conectarea la rețele WiFi, gestionarea conexiunilor și manipularea diverselor evenimentelor de rețea. Totodată, ea permite recuperarea informațiilor despre rețea curentă, cum ar fi SSID-ul, adresa IP, adresa MAC sau intensitatea semnalului. Din punct de vedere al securității, suportă diverse protocoale de securitate WiFi, inclusiv WEP, WPA și WPA2, pentru a asigura conexiuni sigure și protejate [37].

2.3. Biblioteca NimBLEDevice

Biblioteca NimBLEDevice.h oferă o metodă de implementare eficientă a protocolului Bluetooth Low Energy (BLE). Biblioteca facilitează comunicația wireless între dispozitive prin intermediul BLE, fiind ideală pentru proiecte IoT [38].

Caracteristici cheie:

- *Initializarea Dispozitivului BLE*: Oferă funcții pentru inițializarea dispozitivului BLE și configurarea setărilor acestuia, cum ar fi nivelul de putere.
- *Funcționalitate Client și Server*: Biblioteca suportă operațiuni atât de client, cât și de server BLE, permitând dispozitivelor să anunțe servicii și caracteristici sau să se conecteze la alte dispozitive BLE pentru a interacționa cu serviciile acestora.
- *Managementul Serviciilor și Caracteristicilor*: Permite crearea și gestionarea serviciilor și caracteristicilor BLE, care definesc datele și operațiunile expuse de dispozitiv.
- *Gestionarea Evenimentelor*: Biblioteca oferă mecanisme pentru gestionarea diverselor evenimente BLE, cum ar fi conexiuni, deconectări și operațiuni de citire/scriere a caracteristicilor.
- *Funcționalități de Securitate*: Suportă funcționalități de securitate BLE, inclusiv împerecherea și bonding-ul, asigurând o comunicație securizată între dispozitive.

2.4. Detalierea Codului Sursă

În cadrul codului sursă am definit următoarele macrouri:

- WIFI_SSID: numele rețelei WiFi la care dispozitivul trebuie să se conecteze.
- WIFI_PASS: parola rețelei WiFi.
- APP_KEY: cheia aplicației pentru autentificare la un serviciu cloud.
- APP_KEY si APP_SECRET: elemente necesare pentru autentificarea la un serviciu cloud.
- LIGHT_ID: ID-ul dispozitivului care urmează să fie controlat (în cazul meu ghirlanda luminoasă).

La nivel global, sunt declarate un obiect de tip NimBLEAddress (serverAddress), două obiecte de tip NimBLEUUID (serviceUUID și characteristicUUID) și câte un pointer null, de tipul NimBLEClient (pClient), NimBLERemoteService (pService) și NimBLERemoteCharacteristic (pCharacteristic), liniile 7-13.

Tot aici, am declarat și doi vectori de tip uint8_t ce sunt utilizati în mesajele ce controlează ghirlanda luminoasă, liniile 15-16.

```

1 #define WIFI_SSID      "Nume"      // numele retelei
2 #define WIFI_PASS     "Parola"    // parola de acces
3 #define APP_KEY        "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" // App Key
4 #define APP_SECRET     "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx-xxx..." // App Secret
5 #define LIGHT_ID       "Id_dizpozitiv" // Device Id
6
7 static NimBLEAddress serverAddress ("74:d2:85:5e:40:34"); //adresa MAC a ghirlandei

```

```

8 static NimBLEUUID serviceUUID("1000");
9 static NimBLEUUID characteristicUUID("1001");
10
11 NimBLEClient* pClient = nullptr;
12 NimBLERemoteService* pService = nullptr;
13 NimBLERemoteCharacteristic* pCharacteristic = nullptr;
14
15 uint8_t rgb_color_data[] = { 0x3c, 0x02, 0x00, 0x00, 0x00 };
16 uint8_t on_off_data[] = { 0x3c, 0x01, 0x00 }; //on 3c0101, off 3c0100

```

Funcția **setupWiFi()** este utilizată pentru a stabili conexiunea WiFi stabilă.

Metodele WiFi.setSleep(false) și WiFi.setAutoReconnect(false) sunt utilizate pentru a dezactiva modul de economisire a energiei și, respectiv, pentru a activa reconectarea automată, liniile 2-3.

WiFi.begin(WIFI_SSID, WIFI_PASS) este folosit pentru începerea conectării la rețea, iar WiFi.status() pentru a verifica dacă dispozitivul s-a conectat cu succes. Structura repetitivă while, liniile 7-10, a fost adăugată cu scopul de a ”urmări” progresul conectării (printează câte un punct la un interval de 500 ms cât timp statusul este diferit de WL_CONNECTED, adică atât timp cât dispozitivul nu este conectat încă la Wi-Fi).

Odată realizată conexiunea, mesajul ”WiFi connected” este afișat în serial monitor, urmat de mesajul ”IP address: ” și de adresa IP a dispozitivului. Adresa a fost afișată cu ajutorul WiFi.localIP(), care returnează adresa IP locală a dispozitivului, linia 15.

```

1 void setupWiFi() {
2     Serial.printf("\r\n[Wifi]: Connecting");
3     WiFi.setSleep(false);
4     WiFi.setAutoReconnect(true);
5     WiFi.begin(WIFI_SSID, WIFI_PASS);
6
7     while (WiFi.status() != WL_CONNECTED) {
8         Serial.printf(".");
9         delay(500);
10    }
11
12    Serial.println("");
13    Serial.println("WiFi connected");
14    Serial.println("IP address: ");
15    Serial.println(WiFi.localIP());
16 }

```

Funcția **disconnectWiFi()** deconectează microcontrolerul de la rețea și eliberează resursele cu ajutorul metodei WiFi.disconnect(true). Delay-ul de 500ms asigură că a fost alocat suficient timp pentru ca resursele să fie eliberate. După deconectare, mesajul ”WiFi disconnected” este afișat.

```

1 void disconnectWiFi() {
2     WiFi.disconnect(true);
3     delay(500);
4     Serial.printf("WiFi disconnected");
5 }

```

Funcția **reconnectWiFi()** este utilizată pentru reconectarea la rețeaua WiFi. Similar cu setupWiFi(), ”urmărește” progresul conectării prin printarea unor puncte la interval de 500ms până când conexiunea este stabilită, liniile 5-8, și apoi afișează mesajul ”WiFi reconnected”, linia 11.

```

1 void reconnectWiFi() {
2     Serial.printf("\r\n[Wifi]: Connecting");
3     WiFi.begin(WIFI_SSID, WIFI_PASS);
4
5     while (WiFi.status() != WL_CONNECTED) {
6         Serial.printf(".");
7         delay(500);
8     }
9
10    Serial.printf("");
11    Serial.printf("\n WiFi reconnected\n");
12 }
```

Scopurile funcției **disconnectBLE()** sunt deconectarea de la serverul BLE și eliberarea resurselor. Metoda pClient->disconnect() a fost chemată pentru a realiza deconectarea, iar NimBLEDevice::deleteClient(pClient) și NimBLEDevice::deinit() pentru a elibera resursele, liniile 5-6.

Pentru a preveni viitoare probleme, înainte ca funcțiile onPowerState() sau onColor() să poată fi chemate din nou, pointerilor pClient, pService și pCharacteristic li se realocă valoarea nullptr și astfel ne asigurăm că nu rămân pastrate adresele de memorie anterioare, liniile 7-9.

În urma deconectării și eliberării resurselor, mesajul "BLE disconnected and resources freed" este afișat pe serial monitor.

```

1 void disconnectBLE() {
2     pClient->disconnect();
3     delay(500);
4
5     NimBLEDevice::deleteClient(pClient); // Delete the client object
6     NimBLEDevice::deinit(); // Deinitialize NimBLE
7     pClient = nullptr; // Reset pointer
8     pService = nullptr; // Reset pointer
9     pCharacteristic = nullptr; // Reset pointer
10    Serial.println("BLE disconnected and resources freed");
11 }
```

Funcția **setupSinricPro()** configurează biblioteca SinricPro pentru a gestiona un dispozitiv de tip Smart Light (lumină inteligentă). Aceasta începe prin a seta de funcții callback care gestionează schimbările stării de alimentare și de culoare cu ajutorul metodelor myLight.onPowerState(onPowerState) și myLight.onColor(onColor), liniile 6-7.

Apoi, configuraază SinricPro.onConnected și SinricPro.onDisconnected cu funcții callback care sunt apelate atunci când dispozitivul se conectează și se deconectează de la cloud-ul SinricPro, liniile 10-11. În cazul meu, am folosit cele două funcții pentru a primi feedback prin serial monitor legat de starea conexiunii.

La final, initializează biblioteca SinricPro cu credențialele necesare pentru conectarea la cloud-ul SinricPro, cu ajutorul metodei SinricPro.begin(APP_KEY , APP_SECRET), linia 13.

Această funcție este esențială pentru stabilirea comunicării între microcontroller și platforma SinricPro, permitând controlul și automatizarea de la distanță.

```

1 void setupSinricPro() {
2
3     SinricProLight &myLight = SinricPro[LIGHT_ID];
4
5     // set callback function to device
6     myLight.onPowerState(onPowerState);
7     myLight.onColor(onColor);
8 }
```

```

9 // setup SinricPro
10 SinricPro.onConnected([](){ Serial.printf("Connected to SinricPro\r\n"); });
11 SinricPro.onDisconnected([](){ Serial.printf("Disconnected from SinricPro\r\n"); });
12
13 SinricPro.begin(APP_KEY, APP_SECRET);
14 }

```

Funcția **onPowerState()** este un callback utilizat pentru a gestiona starea de alimentare a ghirlandei luminoase prin SinricPro. Această funcție se ocupă de comutarea stării dispozitivului între stările On (pornit) și Off (oprit), trimițând mesaje de tip write command. Datele transmise în mesaj includ vectorul `on_off_data`, iar factorul ce determină schimbarea stării este setarea celui de-al treilea octet din vector (0x01 pentru On și 0x00 pentru Off).

Interpretarea comenzi vocală transmise este accesată prin intermediul parametrului de intrare `state`. Astfel, prin operația de la linia 4, dacă valoarea lui `state` este true, atunci valoarea celui de-al treilea octet din vectorul `on_off_data` devine 0x01, iar dacă `state` este false devine 0x00.

Tot această funcție gestionează și conexiunile Wi-Fi și BLE și comutarea între ele. Deoarece pentru formarea conexiunilor Wi-Fi și BLE sunt folosite resurse comune, precum antena, este necesară întreruperea conexiunii Wi-Fi atunci când vrem să comunicăm prin BLE și invers. În acest scop, pentru gestionarea conexiunii Wi-Fi sunt utilizate funcțiile `disconnectWiFi()`, linia 6, și `reconnectWiFi()`, linia 36, la începutul și finalul funcției `onPowerState()`.

Transmiterea comenzi de la microcontroller către ghirlanda luminoasă în funcția `onPowerState()` implică următorii pași:

1. Afisarea stării în care urmează să fie dispozitivul, linia 3.
2. Prelucrarea interpretării comenzi vocală de la parametrul de intrare `state` și modificarea celui de-al treilea octet din vectorul `on_off_data`, linia 4.
3. Deconectarea de la Wi-Fi, linia 6.
4. Inițializarea dispozitivului BLE și setarea nivelului de putere, liniile 9 și 10.
5. Crearea și conectarea clientului BLE (creează un client BLE și încearcă să se conecteze la serverul BLE specificat prin `serverAddress`, iar dacă conexiunea este reușită, afișează un mesaj de confirmare, liniile 12-14).
6. Obținerea serviciului și a caracteristicii BLE (obține serviciul BLE specificat de `serviceUUID` și caracteristica specificată de `characteristicUUID` de la serverul BLE, liniile 16-18).
7. Scrierea valorii caracteristicii, liniile 19-26:
 - dacă a fost gasita și este disponibilă, adică dacă `pCharacteristic` este diferit de null, încearcă să scrie `on_off_data` în caracteristică.
 - afișează un mesaj de succes sau eșec în funcție de rezultatul scrierii
 - dacă scrierea a avut succes, introduce o întârziere de 500 ms, asigurându-se astfel ca mesajul are timp să ajungă la și să fie procesat de ghirlanda luminoasă, înainte de a transmite urmatorul mesaj
8. Deconectarea BLE și reconectarea la Wi-Fi, liniile 34-35.

```

1 bool onPowerState(const String &deviceId, bool &state) {
2
3     Serial.printf("Device turned %s\n", state?"on":"off");
4     on_off_data[2] = state ? 0x01 : 0x00;
5     //disconnect from wifi and turn off the wifi radio
6     disconnectWiFi();
7
8     //initialize BLE
9     NimBLEDevice::init("");
10    NimBLEDevice::setPower(ESP_PWR_LVL_P9);
11    // Connect to the bt server
12    pClient = NimBLEDevice::createClient();
13    if (pClient->connect(serverAddress)) {
14        Serial.println("Connected to server");
15        // Get the service and characteristic
16        pService = pClient->getService(serviceUUID);
17        if (pService){
18            pCharacteristic = pService->getCharacteristic(characteristicUUID);
19            if ( pCharacteristic ) {
20                if (pCharacteristic->writeValue(on_off_data,sizeof(on_off_data), 0)) {
21                    Serial.println("Write successful");
22                    delay(500);
23                } else {
24                    Serial.println("Write failed");
25                }
26            }
27        }
28    } else {
29        Serial.println("Failed to connect to server");
30    }
31
32    //disconnect BLE
33    disconnectBLE();
34    reconnectWiFi();
35    return true;
36}
37

```

Funcția **onColor()** este un callback utilizat pentru a gestiona culoarea ghirlandei luminoase. Implementarea transmiterii comenzi de la microcontroller către ghirlandă este aproape identică cu cea din **onPowerState()**. Principalele diferențe apar în etapa de prelucrare a interpretării comenzi vocale și în vectorul trimis prin write command.

Spre deosebire de **onPowerState()**, unde comanda era transmisă printr-un singur parametru (state), în **onColor()**, comanda este "tradusă" în trei parametri de intrare (r - care corespunde componentei roșu, g - care corespunde componentei verde și b - care corespunde componentei albastru).

Vectorul transmis către ghirlandă luminoasă în **onColor()** este un vector cu cinci elemente (**rgb_color_data**), comparativ cu cel din **onPowerState()**, care avea doar trei elemente (**on_off_data**). În cazul funcției **onColor()**, cele trei valori r, g și b corespondente input-ului sunt stocate pe pozițiile celor de-al treilea, al patrulea și respectiv al cincilea octet, liniile 2-4.

Transmiterea comenzi de la microcontroller către ghirlandă luminoasă în funcția **onColor()** implică următorii pași:

1. Prelucrarea interpretării comenzi vocale, liniile 2-4.
2. Afisarea componentelor RGB ale culorii ce urmează să fie trimisă dispozitivului, linia 5.
3. Deconectarea de la Wi-Fi, linia 8.

4. Inițializarea dispozitivului BLE și setarea nivelului de putere, liniile 10 și 11.
5. Crearea și conectarea clientului BLE, liniile 13-15.
6. Obținerea serviciului și a caracteristicii BLE, liniile 17-19.
7. Scrierea valorii caracteristicii, liniile 20-27.
8. Deconectarea BLE și reconectarea la Wi-Fi, liniile 35-36.

```

1 bool onColor(const String &deviceId, byte &r, byte &g, byte &b) {
2     rgb_color_data[2] = r;
3     rgb_color_data[3] = g;
4     rgb_color_data[4] = b;
5     Serial.printf("Device color changed to %d, %d, %d (RGB)\r\n", rgb_color_data[2],
6                   rgb_color_data[3], rgb_color_data[4]);
7
8     //disconnect from wifi
9     disconnectWiFi();
10    //initialize BLE
11    NimBLEDevice::init("");
12    NimBLEDevice::setPower(ESP_PWR_LVL_P9);
13    // Connect to the ble server
14    pClient = NimBLEDevice::createClient();
15    if (pClient->connect(serverAddress)) {
16        Serial.println("Connected to server");
17        // Get the service and characteristic
18        pService = pClient->getService(serviceUUID);
19        if (pService){
20            pCharacteristic = pService->getCharacteristic(characteristicUUID);
21            if ( pCharacteristic ) {
22                if (pCharacteristic->writeValue(rgb_color_data, sizeof(rgb_color_data), 0)) {
23                    Serial.println("Write successful");
24                    delay(500);
25                } else {
26                    Serial.println("Write failed");
27                }
28            }
29        }
30    } else {
31        Serial.println("Failed to connect to server");
32    }
33
34    //disconnect BLE
35    disconnectBLE();
36    reconnectWiFi();
37    return true;
38 }
```

Metoda **SinricPro.handle ()** este esențială pentru funcționarea corectă a bibliotecii SinricPro. Aceasta este apelată într-un loop pentru a asigura gestionarea eficientă a evenimentelor și a comunicațiilor între dispozitiv și serviciul cloud SinricPro.

Metoda **Serial.begin(115200)** initializează portul serial și setează viteza de baudrate specificată (de exemplu, 9600 bps). Aceasta pregătește portul serial pentru comunicare.

3. Obstacole întâlnite și soluționarea lor

3.1. Alegerea bibliotecilor folosite

Pentru a oferi o soluție capabilă de conectivitate la Wi-Fi și BLE, care să permită controlul ghirlandei luminoase prin comenzi vocale cu ajutorul Amazon Alexa / Google Home, am utilizat biblioteci specializate pentru fiecare dintre funcționalitățile oferite.

În procesul de realizare a proiectului, am aplicat abordarea ”divide et impera”, adică am ales să implementez inițial două versiuni, fiecare dintre ele suportând funcționalități diferite ale produsului final.

Prima versiune a fost gândită să suporte conectivitatea la rețea Wi-Fi și capacitatea de control prin comenzi vocale al unui LED conectat direct la placa de dezvoltare, iar cea de-a doua versiune să suporte controlul ghirlandei luminoase prin intermediul BLE. În acest scop, am utilizat bibliotecile WiFi.h, SinricPro.h și SinricProSwitch.h pentru prima versiune și BLEDevice.h pentru a doua versiune.

După ce cele două versiuni au fost implementate cu succes, am încercat să implementez o nouă versiune care să suporte toate funcționalitățile. Însă, din păcate, ceea ce nu luasem în considerare era posibilitatea depășirii capacitatii de stocare a dispozitivului.

Pentru soluționarea acestei probleme, am ales să înlocuiesc biblioteca BLEDevice.h cu NimBLEDevice.h, care a fost dezvoltată special pentru dispozitive cu resurse limitate din punct de vedere al memoriei.

O altă modificare adusă față de primele versiuni a constat în înlocuirea bibliotecii SinricProSwitch.h cu SinricProLight pentru a putea implementa funcționalitatea de schimbare a colorii necesară pentru controlul ghirlandei luminoase RGB.

3.2. Gestionarea resurselor

Un alt obstacol cu care m-am confruntat în realizarea proiectului a fost cauzat de faptul că pentru a fi create și menținute atât conexiunea la Wi-Fi cât și cea la dispozitivul BLE utilizau resurse comune. În consecința celei două tipuri de conexiuni nu pot fi susținute simultan.

Solutia în acest caz a fost realizarea unei implementări în care se conectează pe rand la Wi-Fi sau BLE în funcție de nevoie.

După ce modulul este conectat la alimentare, acesta stabilește o conexiune cu cloud-ul SinricPro prin Wi-Fi și așteaptă să primească o comandă vocală. Odată ce primește comanda, se deconectează de la Wi-Fi, se conectează la ghirlinda luminoasă prin BLE și îi trimite un mesaj pentru a-i schimba starea conform instrucțiunilor primite. După ce mesajul a fost trimis, modulul se deconectează de la BLE, se reconectează la Wi-Fi și apoi așteaptă să primească o nouă comandă vocală. Acest proces se repetă continuu până la deconectarea de la alimentare.

3.3. Redactarea documentației proiectului

Pentru redactarea documentației am folosit Overleaf, o platformă online de editare și colaborare pentru documente L^AT_EX, foarte utilizată în mediul academic și profesional pentru crearea de documente științifice, [39].



Figura 3.1.: Overleaf logo [40]

În timpul redactării, aspectul care a creat cele mai multe probleme a fost poziția imaginilor în document. Atunci când inserezi o imagine, aceasta este așezată în pagină în funcție de locul disponibil. Însă, dacă spațiul rămas pe pagină era insuficient, imaginea era mutată la începutul următoarei pagini și înlocuită de textul imediat următor.

Acest lucru cauza confuzie și probleme în înțelegerea conținutului, deoarece unele imagini ajungeau să fie inserate în alte capitole sau secțiuni care nu aveau nicio legătură cu conținutul lor. De asemenea, inserarea într-o locație nepotrivită a unei imagini crea un efect de cascadă, cauzând o mare parte din următoarele imagini să își schimbe și ele poziția.

Pentru soluționarea acestor situații, am modificat repetat dimensiunea imaginii și poziția în care să fie inserată. În unele cazuri, am schimbat ordinea imaginilor sau a secțiunilor de conținut pentru a obține o înșiruire a informațiilor coerentă și corectă.

4. Evaluarea Funcționalității Proiectului

Testarea reprezintă un proces esențial în dezvoltarea software-ului, având scopul de a verifica și valida funcționalitatea, calitatea și performanța produsului software. Prin testare, se identifică și se corectează erorile (bug-urile) înainte ca produsul să fie livrat utilizatorilor finali.

Beneficiile testării sunt:

- *Identificarea erorilor și defectelor:* Testarea ajută la descoperirea erorilor și defectelor în software, prevenind astfel problemele majore în faza de producție.
- *Asigurarea calității software-ului:* Procesul de testare asigură că software-ul respectă specificațiile și cerințele inițiale, garantând astfel calitatea și conformitatea produsului final.
- *Satisfacția utilizatorilor:* Un software bine testat este mai fiabil și oferă o experiență mai bună utilizatorilor finali, crescând astfel satisfacția acestora.
- *Documentarea și îmbunătățirea procesului de dezvoltare:* Testarea oferă informații valoroase despre comportamentul software-ului, care pot fi folosite pentru îmbunătățirea proceselor de dezvoltare viitoare.

În cadrul procesului de testare al proiectului meu IoT, am utilizat intens afișarea de mesaje pe monitorul serial pentru a monitoriza și verifica funcționarea corectă a sistemului. Prin intermediul funcției Serial.printf(), am inserat mesaje informative la fiecare etapă critică a procesului, cum ar fi inițializarea conexiunii WiFi, schimbarea stării dispozitivelor și conectarea/disconectarea la serverul BLE.

În funcția setupWiFi(), am afișat mesaje pentru a indica începutul procesului de conectare, fiecare etapă de verificare a stării conexiunii și succesul final al conectării.

- La începutul funcției, mesajul "[Wifi]: Connecting" este afișat pentru a indica faptul că procesul de conectare WiFi a început.
- În bucla while, se afișează un punct (".") la fiecare 500 ms pentru a vizualiza procesul de conectare în desfășurare. Acest lucru ajută la confirmarea că dispozitivul încearcă să se conecteze la rețea.
- După ce conexiunea este stabilită, mesajele WiFi connected și IP address: sunt afișate, urmate de adresa IP alocată. Aceste mesaje confirmă că dispozitivul s-a conectat cu succes la rețeaua WiFi.

Similar, în funcțiile onPowerState() și onColor(), am inclus mesaje care indică schimbarea stării dispozitivului și rezultatele operațiilor de scriere a caracteristicilor BLE.

- În funcția onPowerState(), mesajul Device turned on sau Device turned off este afișat în funcție de starea setată de comanda primită. Acest mesaj ajută la confirmarea faptului că comanda de pornire/oprire este recepționată și procesată corect.
- În funcția onColor(), mesajul Device color changed to [r, g, b] este afișat pentru a indica modificarea culorii dispozitivului. Aceasta ajută la verificarea că valorile RGB sunt transmise și aplicate corect.

Am inclus, de asemenea, mesaje pentru a verifica funcționarea corectă a funcțiilor de conectare și deconectare WiFi și BLE.

- Mesajele "[Wifi]: Connecting" și "WiFi reconnected" sunt afișate pentru a indica stadiul procesului de reconectare WiFi.
- Mesajul "WiFi disconnected" confirmă că deconectarea de la rețeaua WiFi s-a realizat cu succes.
- Mesajul "BLE disconnected and resources freed" indică faptul că dispozitivul BLE a fost deconectat și resursele au fost eliberate corect.

Prin afișarea acestor mesaje în consola serială, am reușit să monitorizez și să verific corectitudinea proceselor de conectare WiFi, interacțiunile BLE și funcționarea generală a dispozitivului IoT.

Pe lângă testarea prin afișarea de mesaje în consola serială, am realizat și o testare practică a sistemului prin trimiterea de comenzi de control și verificarea vizuală a efectelor acestora.

Aceaste metode de testare au facilitat identificarea și corectarea problemelor în fazele timpurii ale dezvoltării, asigurând că toate componentele sistemului funcționează corect și sincronizat.

5. Contribuții personale

În cadrul acestui proiect de diplomă, am adus contribuții semnificative în mai multe aspecte, de la dezvoltarea software-ului până la integrarea și testarea dispozitivelor, abordând provocările tehnice cu soluții ingenioase.

Dezvoltarea și implementarea software-ului Prima mea contribuție majoră a fost dezvoltarea unei aplicații software pentru un sistem embedded capabil să comunice atât cu Google Home, cât și cu Amazon Alexa. Pentru a realiza acest lucru, am utilizat bibliotecile WiFi.h, SinricPro.h, SinricProLight.h și NimBLEDevice.h. În cadrul implementării, am integrat funcționalitățile de conectivitate Wi-Fi pentru primirea comenziilor vocale și controlul dispozitivelor Bluetooth Low Energy (BLE) pentru a trimite mai departe comenziile catre corpul de iluminat.

Optimizarea utilizării resurselor Unul dintre obstacolele majore întâlnite a fost limitarea resurselor de memorie ale dispozitivului embedded, care a impus utilizarea unei biblioteci BLE mai eficiente. Am înlocuit biblioteca BLEDevice.h cu NimBLEDevice.h și am rescris codul ce permitea suportul functionalitatii de conectare la un dispozitiv BLE. În adiție am implementat o metodă prin care sistemul să gestioneze eficient conexiunile Wi-Fi și BLE, asigurând funcționarea corectă și stabilă a sistemului .

Integrarea cu SinricPro și Alexa/Google Home Pentru a asigura compatibilitatea cu platformele de asistență vocală, am integrat sistemul cu SinricPro. Aceasta a necesitat configurarea corectă a dispozitivului și utilizarea protocolelor de comunicare adecvate pentru a permite interacțiunea fluidă între dispozitivul BLE și hub-urile de asistență vocală.

Ingineria Inversă a Pachetelor BLE O contribuție esențială a fost realizarea procesului de inginerie inversă a pachetelor BLE pentru a identifica protocolele de comunicație ale dispozitivului de iluminat. Am utilizat instrumente precum Wireshark pentru a captura și analiza pachetele de date, reușind să extrag informațiile necesare pentru a implementa controlul corect al luminilor prin intermediul sistemului embedded. Acest proces a implicat teste extinse pentru a asigura acuratețea și fiabilitatea comunicării .

Redactarea Documentației Tehnice Un alt aspect important al contribuției mele a fost redactarea documentației tehnice detaliate a proiectului. Am folosit platforma Overleaf pentru a crea documentația în format LaTeX, asigurându-mă că toate aspectele tehnice și metodologice sunt clar prezentate și ușor de urmărit.

Concluzii

În concluzie, prin realizarea proiectului de diplomă „Integrare a unui sistem embedded cu Google Home / Amazon Alexa” am reușit să dezvolt o soluție care utilizează atât Wi-Fi, cât și Bluetooth Low Energy (BLE) pentru a controla dispozitivele inteligente de iluminat și care oferă utilizatorilor o experiență intuitivă și convenabilă.

Perspective de dezvoltare în viitor

Lista a unor posibile îmbunatatiri

- *Optimizarea performanței și reducerea latenței:*

Îmbunătățirea algoritmilor de procesare și a structurii rețelei pentru a reduce timpul de răspuns și a asigura o interacțiune mai rapidă și mai eficientă între utilizatori și dispozitive.

- *Extinderea compatibilității cu alte platforme de asistență vocală:*

Integrarea cu alte platforme populare, precum Apple HomeKit sau Samsung SmartThings, pentru a asigura o compatibilitate mai largă și a oferi utilizatorilor mai multe opțiuni de control vocal.

- *Implementarea unor funcționalități avansate de automatizare:*

Dezvoltarea unor scenarii de automatizare complexă, cum ar fi declanșarea unor acțiuni bazate pe evenimente specifice (de exemplu, activarea unor lumini la detectarea mișcării sau la o anumită oră).

- *Monitorizarea și raportarea stării dispozitivelor:*

Implementarea unui sistem de monitorizare în timp real care să ofere utilizatorilor informații despre starea și performanța dispozitivelor.

Cateva exemple ar putea fi:

- Afisare informații precum statusul conexiunii Wi-Fi (conectat/deconectat), puterea semnalului (RSSI), adresa IP.
- Avertismente în cazul pierderii conexiunii Wi-Fi sau semnalului slab.
- Afisare informații precum statusul conexiunii BLE (conectat/deconectat), nivelul bateriei dispozitivului (dacă este disponibil), ultimele comenzi executate.
- Alarme în cazul în care conexiunea BLE este pierdută sau nivelul bateriei este scăzut.

- *Adăugarea suportului pentru noi tipuri de dispozitive IoT:*

Extinderea compatibilității sistemului cu alte tipuri de dispozitive inteligente, cum ar fi termostate sau senzori de mediu, pentru a oferi o automatizare completă a locuinței.

- *Feedback și învățare automată:*

Implementarea unor mecanisme de feedback și învățare automată pentru a adapta și optimiza funcționarea sistemului în funcție de preferințele și comportamentele utilizatorului.

Un exemplu concret ar fi ajustarea automată a nuanței de roz în funcție de preferințele utilizatorului. Sistemul va înregistra și analiza comenziile utilizatorului pentru a identifica tipare. Dacă detectează că utilizatorul ajustează în mod repetat culoarea "roz" la un "roz mai deschis" de mai multe ori, va considera acest comportament ca o preferință.

Bibliografie

- [1] *Introducing JSON*, <https://www.json.org/json-en.html>, Accessed: 2024-06-13.
- [2] Ian Fette și Alexey Melnikov, *The WebSocket Protocol*, RFC Editor, 2011, URL: <https://datatracker.ietf.org/doc/html/rfc6455>.
- [3] Somayya Madakam, R. Ramaswamy și S. Tripathi, „Internet of Things (IoT): A Literature Review“, în *Journal of Computer and Communications* 3 (2015), pp. 164–173, DOI: [10.4236/jcc.2015.35021](https://doi.org/10.4236/jcc.2015.35021).
- [4] Biljana L. Risteska Stojkoska și Kire V. Trivodaliev, „A review of Internet of Things for smart home: Challenges and solutions“, în *Journal of Cleaner Production* 140 (2017), pp. 1454–1464, ISSN: 0959-6526, DOI: <https://doi.org/10.1016/j.jclepro.2016.10.006>, URL: <https://www.sciencedirect.com/science/article/pii/S095965261631589X>.
- [5] Philips, *Philips hue*, <https://www.philips-hue.com/en-my/p/hue-bridge/8719514342644>, Accessed: 2024.06.10, 2024.
- [6] C. Muthu Ramya, M Shanmugaraj și R Prabakaran, „Study on ZigBee technology“, în *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, 2011, pp. 297–301, DOI: [10.1109/ICECTECH.2011.5942102](https://doi.org/10.1109/ICECTECH.2011.5942102).
- [7] Seeed Studio, *XIAO ESP32-C3 Getting Started*, https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/, Accessed: 2024.06.10, 2024.
- [8] Espressif, *ESP32-C3 Datasheet*, https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf, Accessed: 2024.06.10, 2024.
- [9] Seeed Studio, *XIAO ESP32-C3 WiFi usage*, https://wiki.seeedstudio.com/XIAO_ESP32C3_WiFi_Usage/, Accessed: 2024.06.10, 2024.
- [10] HK Zhengda, *Actuel RGB light*, <https://play.google.com/store/apps/details?id=com.ttcble.zhenda.actuel.rgb&hl=ro&gl=US>, Accessed: 2024.06.10, 2024.
- [11] Wi-Fi Alliance, *Who We Are: Our Brands*, 2024, URL: <https://www.wi-fi.org/who-we-are/our-brands>.
- [12] K. Pahlavan și P. Krishnamurthy, „Evolution and Impact of Wi-Fi Technology and Applications: A Historical Perspective“, în *Int J Wireless Inf Networks* 28 (2021), pp. 3–19, DOI: [10.1007/s10776-020-00501-8](https://doi.org/10.1007/s10776-020-00501-8), URL: <https://doi.org/10.1007/s10776-020-00501-8>.
- [13] Jaidev Sharma, „The Wi-Fi Evolution“, în *Online; accessed. Sep.](cited on page) ()*, URL: <https://www.kufunda.net/publicdocs/qorvo-the-wi-fi-evolution-white-paper.pdf>.
- [14] Cailian Deng et al., „IEEE 802.11be Wi-Fi 7: New Challenges and Opportunities“, în *IEEE Communications Surveys Tutorials* 22.4 (2020), pp. 2136–2166, DOI: [10.1109/COMST.2020.3012715](https://doi.org/10.1109/COMST.2020.3012715).
- [15] Cisco, *How Does a Router Work?*, Accessed: 2024-06-22, 2024, URL: <https://www.cisco.com/c/en/us/solutions/small-business/resource-center/networking/how-does-a-router-work.html#~how-routers-help>.
- [16] TechTarget, *Service Set Identifier (SSID)*, Accessed: 2024-06-22, 2024, URL: <https://www.techtarget.com/searchmobilecomputing/definition/service-set-identifier>.
- [17] Jeffrey G. Andrews, Arogyaswami Ghosh și Rias Muhammed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*, Prentice Hall, 2007.

- [18] Saurabh Malgaonkar et al., „Research on Wi-Fi Security Protocols“, în *International Journal of Computer Applications* 164.3 (2017), pp. 30–36.
- [19] Elyas Baray și Nitish Kumar Ojha, „WLAN Security Protocols and WPA3 Security Approach Measurement Through Aircrack-ng Technique“, în *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, pp. 23–30, DOI: [10.1109/ICCMC51019.2021.9418230](https://doi.org/10.1109/ICCMC51019.2021.9418230).
- [20] B Indira Reddy și V Srikanth, „Review on wireless security protocols (WEP, WPA, WPA2 & WPA3)“, în *International Journal of Scientific Research in Computer Science, Engineering and Information Technology* 5.4 (2019), pp. 28–35.
- [21] Philipp Ebbecke, *Protected Management Frames Enhance Wi-Fi Network Security*, Accessed: 2024-06-22, 2024, URL: <https://www.wi-fi.org/beacon/philipp-ebbecke/protected-management-frames-enhance-wi-fi-network-security>.
- [22] Bluetooth Special Interest Group, *Bluetooth Technology Origin Story*, <https://www.bluetooth.com/about-us/bluetooth-origin/>, Accessed: 2024-06-14.
- [23] SherAli Zeadally, Farhan Siddiqui și Zubair Baig, „25 Years of Bluetooth Technology“, în *Future Internet* 11.9 (2019), ISSN: 1999-5903, DOI: [10.3390/fi11090194](https://doi.org/10.3390/fi11090194), URL: <https://www.mdpi.com/1999-5903/11/9/194>.
- [24] Jaycon Systems, *Bluetooth Technology: What Has Changed Over the Years*, <https://medium.com/jaycon-systems/bluetooth-technology-what-has-changed-over-the-years-385da7ec7154>, Accessed: 2024-06-14, 2019.
- [25] Bluetooth Special Interest Group, *The Difference Between Classic Bluetooth and Bluetooth Low Energy*, <https://www.bluetooth.com/blog/the-difference-between-classic-bluetooth-and-bluetooth-low-energy/>, Accessed: 2024-06-14.
- [26] P. Consani, „Classic Bluetooth vs. Bluetooth Low Energy (BLE)“, în *IoT Lab - Tertium Cloud* (2020), URL: <https://iotlab.tertiumcloud.com/2020/08/19/classic-bluetooth-vs-bluetooth-low-energy-ble/>.
- [27] Microchip Technology Inc., *Bluetooth Architecture - Physical Layer*, <https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/>, Accessed: 2024-06-14.
- [28] Robin Heyden, *Bluetooth Low Energy: The Developer's Handbook*, Prentice Hall, 2012.
- [29] Microchip Technology Inc., *Bluetooth Generic Attribute Profile (GATT) - Attributes*, Accessed: 2024-06-22, 2024, URL: <https://developerhelp.microchip.com/xwiki/bin/view/applications/ble/introduction/bluetooth-architecture/bluetooth-host-layer/bluetooth-generic-attribute-profile-gatt/Attributes/>.
- [30] Nordic Semiconductor, *BLE Characteristics, a beginner's tutorial*, Accessed: 2024-06-22, 2024, URL: <https://devzone.nordicsemi.com/guides/short-range-guides/b/bluetooth-low-energy/posts/ble-characteristics-a-beginners-tutorial>.
- [31] SinricPro, *SinricPro: Smart Home IoT*, <https://help.sinric.pro>, Accessed: 2024-06-13.
- [32] *What is Reverse Engineering?*, <https://www.techopedia.com/definition/3212/reverse-engineering>, Accessed: 2024-06-13.
- [33] Android Developers, *Debugging Bluetooth with Bluetooth HCI Snoop Log*, 2021, URL: <https://developer.android.com/studio/debug/dev-options#bluetooth>.
- [34] Android Developers, *Android Debug Bridge (adb)*, 2021, URL: <https://developer.android.com/studio/command-line/adb>.
- [35] Wireshark Foundation, *Wireshark*, <https://www.wireshark.org/>, Accessed: 2024-06-13.
- [36] Nordic Semiconductor, *nRF Connect for Mobile*, <https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-mobile>, Accessed: 2024-06-13.

- [37] Espressif Systems, *WiFi Library*, <https://github.com/espressif/arduino-esp32/blob/master/README.md>, Accessed: 2024-06-22.
- [38] h2zero, *NimBLE-Arduino GitHub Repository*, <https://github.com/h2zero/NimBLE-Arduino>, Accessed: 2024-06-22.
- [39] Overleaf, *Overleaf Features Overview*, <https://www.overleaf.com/about/features-overview>, Accessed: 2024-06-22, 2024.
- [40] Overleaf, *Overleaf Partner Logos*, <https://www.overleaf.com/for/partners/logos>, Accessed: 2024-06-22, 2024.

Anexe

A. Fișiere surșă

```
1 #include <WiFi.h>
2 #include "SinricPro.h"
3 #include "SinricProLight.h"
4 #include <NimBLEDevice.h>
5
6 #define WIFI_SSID      "Nume"      // numele retelei
7 #define WIFI_PASS      "Parola"    // parola de acces
8 #define APP_KEY        "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" // App Key
9 #define APP_SECRET     "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx-xxx..." // App Secret
10 #define LIGHT_ID       "Id_dizpozitiv" // Device Id
11
12 static NimBLEAddress serverAddress ("74:d2:85:5e:40:34"); //adresa MAC a ghirlandei
13 static NimBLEUUID serviceUUID("1000");
14 static NimBLEUUID characteristicUUID("1001");
15
16 NimBLEClient* pClient = nullptr;
17 NimBLERemoteService* pService = nullptr;
18 NimBLERemoteCharacteristic* pCharacteristic = nullptr;
19
20 uint8_t rgb_color_data[] = { 0x3c, 0x02, 0x00, 0x00, 0x00 };
21 uint8_t on_off_data[] = { 0x3c, 0x01, 0x00 }; //on 3c0101, off 3c0100
22
23 bool onPowerState(const String &deviceId, bool &state) {
24
25     Serial.printf("Device turned %s\n", state?"on":"off");
26     on_off_data[2] = state ? 0x01 : 0x00;
27     //disconnect from wifi and turn off the wifi radio
28     disconnectWiFi();
29
30     //initialize BLE
31     NimBLEDevice::init("");
32     NimBLEDevice::setPower(ESP_PWR_LVL_P9);
33     // Connect to the bt server
34     pClient = NimBLEDevice::createClient();
35     if (pClient->connect(serverAddress)) {
36         Serial.println("Connected to server");
37         // Get the service and characteristic
38         pService = pClient->getService(serviceUUID);
39         if (pService){
40             pCharacteristic = pService->getCharacteristic(characteristicUUID);
41             if ( pCharacteristic ) {
42                 if (pCharacteristic->writeValue(on_off_data,sizeof(on_off_data), 0)) {
43                     Serial.println("Write successful");
44                     delay(500);
45                 } else {
46                     Serial.println("Write failed");
47                 }
48             }
49         }
50     } else {
51         Serial.println("Failed to connect to server");
52     }
53
54     //disconnect BLE
55     disconnectBLE();
```

```
57     reconnectWiFi();
58     return true;
59 }
60
61 bool onColor(const String &deviceId, byte &r, byte &g, byte &b) {
62     rgb_color_data[2] = r;
63     rgb_color_data[3] = g;
64     rgb_color_data[4] = b;
65     Serial.printf("Device color changed to %d, %d, %d (RGB)\r\n", rgb_color_data[2],
66                   rgb_color_data[3], rgb_color_data[4]);
67
68     //disconnect from wifi
69     disconnectWiFi();
70     //initialize BLE
71     NimBLEDevice::init("");
72     NimBLEDevice::setPower(ESP_PWR_LVL_P9);
73     // Connect to the ble server
74     pClient = NimBLEDevice::createClient();
75     if (pClient->connect(serverAddress)) {
76         Serial.println("Connected to server");
77         // Get the service and characteristic
78         pService = pClient->getService(serviceUUID);
79         if (pService){
80             pCharacteristic = pService->getCharacteristic(characteristicUUID);
81             if ( pCharacteristic ) {
82                 if (pCharacteristic->writeValue(rgb_color_data, sizeof(rgb_color_data), 0)) {
83                     Serial.println("Write successful");
84                     delay(500);
85                 } else {
86                     Serial.println("Write failed");
87                 }
88             }
89         }
90     } else {
91         Serial.println("Failed to connect to server");
92     }
93
94     //disconect BLE
95     disconnectBLE();
96     reconnectWiFi();
97     return true;
98 }
99
100 void setupWiFi() {
101     Serial.printf("\r\n[Wifi]: Connecting");
102     WiFi.setSleep(false);
103     WiFi.setAutoReconnect(true);
104     WiFi.begin(WIFI_SSID, WIFI_PASS);
105
106     while (WiFi.status() != WL_CONNECTED) {
107         Serial.printf(".");
108         delay(500);
109     }
110
111     Serial.println("");
112     Serial.println("WiFi connected");
113     Serial.println("IP address: ");
114     Serial.println(WiFi.localIP());
115
116 }
117
118 void reconnectWiFi() {
119     Serial.printf("\r\n[Wifi]: Connecting");
120     WiFi.begin(WIFI_SSID, WIFI_PASS);
121 }
```

```
122     while (WiFi.status() != WL_CONNECTED) {
123         Serial.printf(".");
124         delay(500);
125     }
126
127     Serial.printf("");
128     Serial.printf("\n WiFi reconnected\n");
129 }
130
131 void disconnectWiFi() {
132     WiFi.disconnect(true);
133     delay(500);
134     Serial.printf("WiFi disconnected");
135 }
136
137 void disconnectBLE() {
138     pClient->disconnect();
139     delay(500);
140
141     NimBLEDevice::deleteClient(pClient); // Delete the client object
142     NimBLEDevice::deinit(); // Deinitialize NimBLE
143     pClient = nullptr; // Reset pointer
144     pService = nullptr; // Reset pointer
145     pCharacteristic = nullptr; // Reset pointer
146     Serial.println("BLE disconnected and resources freed");
147 }
148
149 // setup function for SinricPro
150 void setupSinricPro() {
151
152     SinricProLight &myLight = SinricPro[LIGHT_ID];
153
154     // set callback function to device
155     myLight.onPowerState(onPowerState);
156     myLight.onColor(onColor);
157
158     // setup SinricPro
159     SinricPro.onConnected([](){ Serial.printf("Connected to SinricPro\r\n"); });
160     SinricPro.onDisconnected([](){ Serial.printf("Disconnected from SinricPro\r\n");
161     });
162
163     SinricPro.begin(APP_KEY, APP_SECRET);
164 }
165
166 void setup() {
167     // initialize digital pin led as an output
168     Serial.begin(115200); Serial.printf("\r\n\r\n");
169     setupWiFi();
170     setupSinricPro();
171 }
172
173 void loop() {
174     if (WiFi.status() == WL_CONNECTED) {
175         SinricPro.handle();
176     } else {
177         Serial.printf("Failed to reconnect to WiFi");
178     }
}
```

Listarea A.1: Codul sursă