UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

COMUNICAREA ÎNTRE NODURI WIRELESS CU PROTOCOLUL ZIGBEE

PROIECT DE DIPLOMĂ

prezentat ca cerință parțială pentru obținerea titlului de *Inginer în domeniul Electronică Aplicată și Ingineria Informației*

Programul de studii *Ingineria Informației*

Conducător științific
Prof. univ. Dr. Ing. Burileanu Corneliu

Absolvent
Călin Horia-Alexandru

București
2017

Anexa 1

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicatii şi Tehnologia Informaţiei
Departamentul: Electronică Aplicată şi Ingineria Informaţiei

Aprobat Director de Departament *:

Prof. Dr. Ing. Sever PAŞCA

TEMA PROIECTULUI DE DIPLOMĂ a studentului Călin Horia-Alexandru, 441A

1. Titlul temei: **"Comunicarea intre noduri wireless cu protocolul Zigbee"**

2. Contribuţia practică, originală a studentului va consta în (*în afara* părţii de documentare):

- *Se va studia modul de funcţionare al nodurilor wireless XBee, ce folosesc protocolul Zigbee iar pe baza acestora se va implementa un sistem de comunicaţie punct-la-punct pe o distanţă de minim 20 de metri. În cadrul procesului de comunicare se va urmări un consum cât mai mic de energie, comparativ cu protocoale wireless asemănătoare: Wi-Fi, Bluetooth etc.*
- *Se vor folosi două sau mai multe plăci de dezvoltare Arduino în vederea facilitării procesului de comunicare şi a interacţiunii cu utilizatorul. Ulterior, la plăci se vor conecta senzori de temperatură, umiditate şi luminozitate, iar informaţiile captate de către acestea vor fi afişate într-o interfaţă grafică.*
- *Se va realiza alimentarea plăcuţelor Arduino în totalitate cu energie solară, folosindu-se panouri solare.*

3. Proiectul se bazează pe cunoştinţe dobândite în principal la următoarele 3-4 discipline: Arhitectura Microprocesoarelor, Microcontrolere, Arhitecturi de Reţea şi Internet, Reţele de Calculatoare,

4. Proprietatea intelectuală asupra proiectului aparţine: *Laborator „SpeeD", UPB, student*

5. Locul de desfăşurare a activităţii: *Laborator „SpeeD", UPB*

6. Realizarea practică rămâne în proprietatea: *Laborator „SpeeD", UPB, student*

7. Data eliberării temei: **15 martie 2016**

CONDUCĂTOR LUCRARE: Prof. Corneliu BURILEANU

STUDENT: Horia-Alexandru CĂLIN

Declarație de onestitate academică

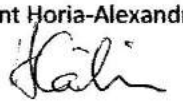
Prin prezenta declar că lucrarea cu titlul "Comunicarea între noduri wireless cu protocolul Zigbee", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică aplicată și Ingineria Informației,* programul de studii *Ingineria Informației* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau o instituție de învățământ 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. 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, *04.09.2016*

Absolvent Horia-Alexandru CĂLIN



CUPRINS

LISTA FIGURILOR

LISTA TABELELOR

LISTĂ DE ACRONIME

A

AP = Access Point

API = Application Programming Interface

B

BLE = Bluetooth Low Energy

BPSK = Binary Phase Shift Keying

BSS = Basic Service Set

C

CPU = Central Processing Unit

CSMA/CD = Carrier-sense Multiple Access/Collision Detection

CSMA/CA = Carrier-sense Multiple Access/Collision Avoidance

D

DSSS = Direct Sequence Spread Spectrum

E

ESS = Extended Servcice Set

F

FFD = Full Functionality Device

I

IBSS = Independent Basic Service Set

IDE = Integrated Development Environment

IoT = Internet of Things

IP = Internet Protocol

O

OPQSK = Offset Quadrature Phase Shift Keying

P

PAN = Personal Area Network

PWM = Pulse Width Modulation

R

RAM = Random Access Memory

ROM = Read Only Memory

RFD = Reduced Functionality Device

T

TDM = Time Division Multiplexing

U

UWB = Ultra Wide Band

W

WAP = Wireless Access Point

Wi-Fi = Wireless Fidelity

WPAN = Wireless Personal Area Network

WSN = Wireless Sensor Networks

CAPITOLUL 1
INTRODUCERE

În cadrul lucrării curente mi-am propus prezentarea generală și explorarea proprietăților protocoalelor de comunicații wireless folosite pentru transmisii de date pe distanțe mici și medii, accentuând avantajele lui ZigBee. Acest protocol va fi analizat și comparat cu altele care îndeplinesc funcții asemănătoare și sunt folosite în prezent în producție. Ulterior voi implementa un număr de trei scenarii în care voi folosi protocolul ZigBee pentru a realiza legătura între componente ce sunt în mod tradițional conectate direct, cu ajutorul firelor.

Necesitatea protocoalelor wireless a crescut exponențial în ultimul deceniu, întrucât tehnologiile curente permit folosirea unui număr foarte mare de dispozitive de monitorizare și control pe o suprafață relativ mică. Prin folosirea unui mod de interconectare standard, cu ajutorul firelor, indiferent dacă vorbim despre fire de cupru sau fibră optică, are loc o aglomerare foarte mare a spațiului de lucru, amănunt care poate conduce către diverse probleme: de natură tehnică, prin apariția diafoniilor între cablurile foarte apropiate sau de natură umană, putând să apară confuzii între cablurile ce trebuie conectate la un anumit pin/interfață/port la un anumit moment de timp pentru a îndeplini o anumită funcționalitate. Deși soluțiile wireless au fost explorate încă de la începutul apariției calculatoarelor personale, doar în trecutul recent s-a pus accentul pe folosirea lor într-un număr mare de situații, datorită necesității rezolvării problemelor de mai sus.

Numărul de aplicații în care protocoalele de comunicații wireless și-au găsit sau își pot găsi aplicabilitatea este destul de mare. Acesta a crescut destul de mult în ultima vreme întrucât s-a căutat optimizarea parametrilor de folosire a mediului wireless: viteza de transmisiune, limitarea interferențelor între canale, dimensiunea pachetelor transmise etc. Chiar dacă cel mai probabil nu se va ajunge niciodată la performanțe egale cu cele ale cablurilor de cupru torsadate, coaxiale sau ale cablurilor ce folosesc fibră optică, soluțiile wireless urmăresc să reprezinte o alternativă viabilă pentru comunicațiile pe distanțe medii, fără a crea inconveniente utilizatorului.

Din punct de vedere practic, implementările realizate în cadrul lucrării prezentate au ca scop evidențierea superiorității protocolului ZigBee în diverse circumstanțe și de asemenea demonstrarea faptului că este un candidat ideal pentru implementarea în tehnologiile Internet of Things. Scopul final al lucrării este acela de a obține mai multe sisteme ce se pot integra cu ușurință într-o locuință ce adoptă tehnologia Internet of Things pentru a conduce către conceptul de ”home automation”.

CAPITOLUL 2
REȚELELE WIRELESS DE NODURI

2.1 EVOLUȚIA REȚELELOR WIRELESS DE NODURI

Conceptul de rețele wireless de noduri se află încă la început, chiar dacă au trecut câțiva ani de când s-a început cercetarea în acest domeniu, respectiv implementarea la nivel mic, crescându-se treptat spre rețele de dimensiuni mai ridicate. Totuși, tendința este aceea de a renunța la nodurile clasice, conectate cu ajutorul firelor și de a migra spre protocoale wireless, mai eficiente din punct de vedere al designului și al implementării. Domeniul în care acest gen de noduri vor avea aplicabilitate este unul foarte vast, existând aplicații dintr-un număr foarte mare de specializări: medicină, supraveghere de securitate, monitorizare mediu înconjurător, militar etc. Prin conectarea într-o rețea a unui număr foarte mare de noduri, obținem posibilitatea să colectăm date legate de fenomene fizice, care ar fi mai greu de monitorizat în moduri convenționale. Probabil, cel mai important domeniu în care această tehnologie va avea aplicabilitate este cel al automatizării – indiferent dacă ne referim la automatizările casnice (pornirea automată a anumitor electrocasnice, deschiderea geamurilor în funcție de diverși parametrii etc.) sau la cele din fabrici (acolo unde poate automatiza întreg procesul de fabricare a diverselor echipamente).

Figura 2.1 – Interconectarea echipamentelor în era modernă [4]

Dacă ritmul prezent al evoluției tehnologiei se va păstra și în următorii ani, iar prețul de fabricație al nodurilor va continua să scadă, atunci ne putem aștepta la rețele care să conțină sute, poate chiar mii de senzori interconectați, având același scop final. După cum vom prezenta în capitolele următoare, ZigBee este singurul protocol capabil să mențină interconectate rețele de dimensiuni atât de mari fără a avea probleme legate de interferențe sau stabilitate a conectivității.

Referindu-ne la diversitatea de aplicații în care își găsesc locul aceste noduri wireless, trebuie luate în considerare și necesitățile ce stau în spatele rețelelor care îi interconectează pe aceștia. Pentru a se îndeplini toate cerințele, de-a lungul timpului au fost propuse mai multe modele de rețele, în jurul cărora au fost construite mai multe modele de protocoale și stive de protocoale. Deși, după cum este exemplificat în cadrul [5] există numeroase moduri în care se poate realiza clasificarea rețelelor de senzori, în continuare scoatem în evidență doar câteva caracteristici ce au un impact major în designul unui protocol.

- Punctele de colectare a datelor (data sink(s)) – în cadrul rețelelor wireless de senzori, avem două tipuri de terminale majore: sursele ce generează date și punctele de colectare, unde datele generate sunt necesare în vederea prelucrării sau trimiterii către sistemele care realizează prelucrarea. În funcție de locația acestor puncte, și intervalul în care acestea au nevoie de date pentru a putea realiza prelucrarea, protocolul trebuie să fie capabil să se adapteze cerințelor.
- Mobilitatea nodurilor - În cazul cel mai general, nodurile sunt presupuse ca fiind imobile. Totuși, după cum vom prezenta și în cadrul părții practice a lucrării, există situații în care trebuie luată în considerare mișcarea senzorilor pentru a satisface nevoile utilizatorilor: exemplul cel mai relevant stă în domeniul roboticii, unde un robot trebuie să fie capabil să ia decizii pe baza senzorilor atașați, sau în domeniul militar, unde necesitățile de supraveghere se schimbă în mod constant. Mobilitatea influențează într-un mod major protocoalele și serviciile de localizare.
- Resursele nodurilor - În funcție de necesitățile de procesare a fiecărui senzor în parte (aici trebuie luate în considerare mărimile măsurate de către aceștia), memoria internă și frecvența de procesare trebuie adaptate corespunzător. Protocoalele pentru rețelele wireless de senzori trebuie să ia în considerare și aceste detalii în momentul în care sunt proiectate.
- Modele de generare a traficului – În majoritatea aplicațiilor, senzorii sunt în cea mai mare parte a timpului într-un mod de ”așteptare”, generând trafic doar atunci când au loc evenimente de interes. În felul acesta se urmărește o eficientizare a consumului de energie. Există totuși și domenii în care este nevoie de monitorizare și generare

continuă de date, cum ar fi supravegherea mediului înconjurător. Evident, între aceste două cazuri nu vor putea fi folosite aceleași protocoale.

După cum se poate observa în cele de mai sus, există un număr mare de criterii ce trebuie luate în considerare în momentul în care este conturat și proiectat un protocol de comunicație.

În continuare ne propunem să prezentăm câteva dintre caracteristicile acestui tip de rețele care oferă avantaje respectiv dezavantaje față de sistemele deja existente în infrastructură.

2.2 CARACTERISTICILE PRINCIPALE ALE REȚELELOR DE NODURI WIRELESS

Precizăm încă de la început că rețelele de senzori au niște caracteristici ce sunt împrumutate de la rețelele ad-hoc, așa cum este prezentat în cadrul lucrării [1]. Totuși, există numeroase diferențe între acestea întrucât cele despre care discutăm în lucrarea curentă sunt mult mai avansate, având o mulțime de funcționalități de care rețele ad-hoc nu dispun.

Totuși, în cadrul proiectării unui protocol pentru rețele de senzori trebuie luate în considerare câteva dintre proprietățile ”moștenite” de la rețelele ad-hoc. Printre acestea se numără:

1. Constrângerile legate de timpul de viață a unui nod, date de către modul de alimentare. În cadrul lucrării prezente mi-am propus alimentarea unor noduri cu ajutorul bateriilor sau a panourilor fotovoltaice, oferind în felul acesta un timp de viață mai ridicat și o mobilitate crescută. Există câteva constrângeri legate de folosirea energiei solare pentru alimentare în general, dar acestea nu se aplică în cazul nostru întrucât senzorii, în general, nu consumă foarte multă energie.
2. Probleme în cadrul procesului de comunicație generate de către mediul prin care se face transmisia wireless. Aici facem referire la interferențele ce pot apărea în mediul în care funcționează nodurile – acestea pot fi cauzate de dispozitive ce transmit de asemenea datele într-o manieră wireless și folosesc o frecvență apropiată de cea a protocolului (cum ar fi telefoanele fixe, care folosesc frecvența de 2.4 GHz pentru conectarea la bază).
3. Abilitatea de auto-configurare, fără a fi necesar un grad de intervenție ridicat din partea administratorilor, odată ce sistemul este configurat și funcțional. În momentul în care apare o problemă în cadrul unei rețele, iar datele nu mai pot fi transmise de la nodurile ce realizează achiziția de date către nodul central, reconfigurarea se face în mod automat, fiind căutate alternative.

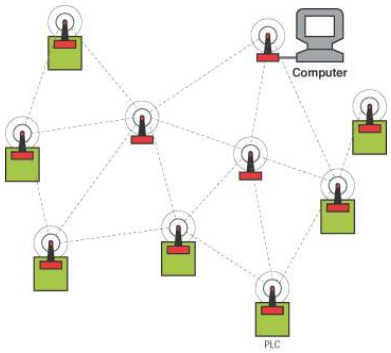


Figura 2.2 – Noduri interconectate wireless [2]

După cum observăm , aceste caracteristici conțin atât avantaje cât și dezavantaje, lăsând deschise numeroase opțiuni de dezvoltare în combaterea dezavantajelor. În cadrul rețelelor de senzori prezentate în lucrarea prezentată, au fost adăugate o serie de caracteristici, prezentate și în [2], la care s-a ajuns plecând de la diverse probleme întâmpinate în implementarea rețelelor ad-hoc.

1. Din punct de vedere al performanțelor, rețele ad-hoc, în forma lor brută nu pot conecta în mod eficient mai mult de 30-40 de senzori, întrucât procesul de comunicație nu este optimizat pentru un număr atât de mare. În WSN-uri, acest lucru este rezolvat de așa natură încât să permită conectarea unui număr de noduri care să ajungă la ordinul miilor.
2. Senzorii în general sunt imobili într-un mediu în care se face monitorizarea diverșilor parametri. Totuși, în cadrul protocoalelor dezvoltate pentru WSN-uri, s-a luat în considerare și posibilitatea deplasării nodurilor, întrucât există domenii în care acest lucru este necesar.
3. Introducerea informațiilor de localizare spațială a nodurilor, relativ la spațiul supravegheat – un detaliu semnificativ pentru mai multe domenii. Ca și exemplu putem considera monitorizarea scurgerilor de gaz – în momentul în care un senzor detectează o astfel de problemă trebuie știut cu exactitate localizarea acestuia.
4. Rețelele de senzori au de cele mai multe ori un model de trafic de tip many-to-one, lucru care poate conduce către probleme de aglomerare a datelor și dificultate în prelucrarea acestora de către nodurile centrale. În WSN-uri aceste probleme se rezolvă prin implementarea mai multor puncte capabile de prelucrare a datelor și cu ajutorul protocoalelor, ce au rolul de a transmite datele către nodul cel mai liber, pentru a nu cauza blocaje. De asemenea, protocolul principal al lucrării, ZigBee, folosește CSMA/CA pentru a evita coliziunile în cadrul rețelelor.
5. Dimensiunile pachetelor de date transmise sunt destul de mici, pentru a simplifica procesul de switching între noduri

În procesul de incorporare a tuturor acestor caracteristici și a multor altele în designul unui protocol, trebuie considerat mereu faptul că resursele fizice ale fiecărui nod sunt limitate și trebuie folosite în mod cât mai eficient. În același timp, protocoalele trebuie să fie foarte simple, pentru a executa operațiile într-un timp cât mai scăzut. Toate aceste reglementări au condus către un număr foarte mare de protocoale dezvoltate pentru a lucra între nivelele acces la rețea și transport, fiecare dintre ele având ca și obiectiv principal proiectarea unei rețele de sine stătătoare, care să opereze un timp cât mai îndelungat fără să fie necesară intervenția unui administrator, păstrând un număr cât mai mare de canale de comunicație și o calitate a serviciilor livrată cât mai ridicată.

2.3 UTILIZAREA SENZORILOR

Deși în cadrul lucrării îmi propun să implementez mai multe scenarii funcționale în care se va folosi un anume protocol wireless de comunicație, ne dorim să demonstrăm de asemenea și utilitatea acestui protocol în rețelele de senzori wireless pentru lucrări viitoare. În continuare vom prezenta câteva caracteristici și proprietăți ale senzorilor și traductorilor, accentuând diferențele dintre acestea în cadrul unui mediu tehnic.

De-a lungul istoriei ne-a fost dovedit în repetate rânduri că progresele făcute atât în știință cât și în inginerie au reprezentat factori majori în dezvoltarea tehnologiei senzorilor. Un prim exemplu din punct de vedere istoric poate să fie reprezentat de sensibilitatea termică a unei rezistențe, observată la începutul anilor 1800 de către Wilhelm von Siemens și implementată ulterior în dezvoltarea unui senzor de temperatură, bazat pe un rezistor de cupru. Stabilitatea cristalelor de cuarț, precum și proprietățile lor piezoelectrice au jucat un rol foarte important în dezvoltarea senzorilor ce sunt folosiți în zilele noastre în aproape orice aplicație, inclusiv în domeniul militar.

Raportându-ne la trecutul mai apropiat, o noua eră în tehnologiile senzorilor a fost declanșată de către posibilitatea explorării proprietăților siliconului în vederea creării unor noi metode de a transforma fenomene fizice în unele electrice, pentru a putea fi citite și procesate de către un calculator. Cercetările ce se realizează în prezent ne vor oferi un control mai bun asupra proprietăților materialelor și a modului în care acestea se comportă, oferindu-ne astfel perspectiva unor noi generații de senzori, mai eficienți și cu un preț de producție redus.

În ciuda faptului ca există, la momentul actual, o cantitate semnificativă de materiale scrise pentru a descrie senzorii și traductorii, se păstrează în continuare o ambiguitate destul de mare în folosirea unei singure definiții pentru aceștia. Motivul este unul destul de simplu: acest domeniu revoluționar, care este într-o continuă dezvoltare este unul interdisciplinar, presupunând cunoștințe din mai multe domenii. De aici deducem că nu ar trebui să ne surprindă inexistența unei definiții acceptate în unanimitate legate de conceptul unui senzor.

Pentru a putea prezenta importanța senzorilor este suficient să ne îndreptăm atenția către dispozitivele electronice pe care le folosim în viața de zi cu zi, atât în mediul de acasă cât și în cel în care ne desfășurăm activitatea profesională, indiferent de domeniu. Atât ei cât și traductorii sunt prezenți peste tot, chiar dacă uneori suntem conștienți de acest lucru întotdeauna, simplificându-ne viața prin eficientizarea activităților întreprinse, de la cele mai semnificative până la cele minore, cărora nu le dăm suficient de multă atenție. Lucrarea de față își propune prezentarea conceptelor generale a acestor senzori și traductori, împreună cu noile tehnologii care înglobează folosirea senzorilor, precum Internet of Things.

Un senzor, în termeni cât mai generali, conform [3], reprezintă o componentă hardware menită să măsoare o cantitate fizică, pe baza căreia emite o un semnal ce poate fi citit și interpretat de către un observator sau un instrument specializat. Unul dintre cele mai simple de înțeles exemple este reprezentat de către termocuplu, care măsoară temperatura la care este expus, iar pe baza unei referințe, emite o tensiune, ce poate fi citită cu ajutorul unui voltmetru. Din punct de vedere al preciziei, toți senzorii sunt calibrați cu ajutorul unor standarde bine cunoscute și definite.

Traductorul reprezintă un dispozitiv ce este alimentat cu energie de către un sistem, în timp ce la rândul său alimentează un al doilea sistem, dar cu energie prezentată sub altă formă. Exemplul cel mai banal este reprezentat de către un difuzor(Figura 1) care transformă semnalele electrice în energie sonoră. Ne putem referi însă și la termocuplul specificat mai sus, capabil să transforme căldura în energie electrică.

În anumite contexte, termenii de senzor și traductori sunt folosiți ca și sinonime, dar acest lucru depinde mult de aplicația în care întâlnim noțiunile.

Senzorii analogici produc un semnal continuu(sau o tensiune), care în general este proporțional cu cantitatea de informație măsurată. Măsurile fizice precum: temperatura, viteza, presiunea etc. reprezintă toate cantități analogice, întrucât sunt de cele mai multe ori continue. Legându-ne la exemplul de mai devreme, temperatura unui lichid poate să fie măsurată cu ajutorul unui termometru sau a unui termocuplu, care răspunde în mod continuu la schimbările de temperatură, pe măsură ce acesta este încălzit sau răcit.

Traductorii ce se află la ieșirea unui sistem sunt denumiți în mod uzual acuatori (dispozitive de acționare) și convertesc energia electrică într-un alt tip de energie, cum ar fi căldură sau energie mecanică.

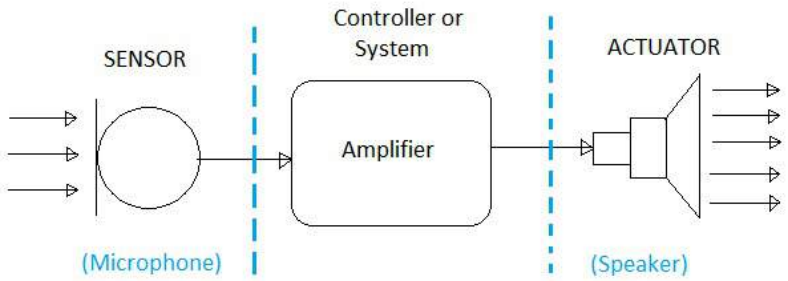


Figura 2.3 – Principiul de funcționare a unui sistem cu senzori [3]

Senzorii digitali au un semnal discret la ieșire, sau o tensiune ce reprezintă cantitatea ce este măsurată. Acest semnal de ieșire poate să fie reprezentat sub forma unui singur bit sau a unei secvențe de biți. Un astfel de senzor este reprezentat, ca și caz general, de către un micro senzor ce are integrat un convertor analog-digital pe o singura pastilă de silicon, pentru a forma o componenta micro-electro-mecanică, ce poate procesa informație sau poate comunica cu un procesor integrat. Aceste caracteristici au dat naștere senzorilor inteligenți din zilele noastre, întâlniți sub denumirea de smart-sensors.

2.4. PROTOCOALE WIRELESS FOLOSITE PENTRU CONECTAREA NODURILOR

La momentul actual, există un număr relativ mare de protocoale wireless ce pot fi folosite pentru diverse aplicații. Acestea sunt prezentate pe larg și detaliat în[4]. În continuare ne propunem să prezentăm o parte dintre aceste protocoale, cele mai folosite și utile din cele existente, făcând o comparație cu protocolul principal al lucrării prezentate – ZigBee.

Protocoalele pe care le vom analiza în continuare în vederea comparației sunt: ZigBee (IEEE 802.15.4), Bluetooth (IEEE 802.15.1), Ultra-Wide-Band (IEEE 802.15.3) și Wi-Fi (IEEE 802.11). Toate cele patru protocoale au un design general pentru rețele wireless de dimensiuni mici, în care toate dispozitivele sunt conectate pe o distanță maximă de câțiva metri – până la câteva zeci. De asemenea, în cadrul proiectării celor patru protocoale, s-a urmărit un consum cât mai mic de energie, pentru a putea oferi dispozitivelor ce folosesc aceste protocoale, o durată de viață a bateriei cat mai mare (sau a unui ciclu de baterie.) Din punct de vedere obiectiv, aplicațiile țintă ale protocoalelor sunt următoarele:

- Bluetooth – tastaturi, mouse fără fir, căști hands-free etc.
- UWB – legături multimedia ce necesită lățime mare de bandă
- ZigBee – orientat pe conexiune, cu un design special pentru rețelele de monitorizare și control a unui număr mare de dispozitive.
- Wi-Fi – conexiunile de tip PC – PC, cu ajutorul unui intermediar (de cele mai multe ori un wireless AP) – vine ca un substituent al rețelelor clasice, ce folosesc cabluri de cupru (coaxiale, torsadate) sau fibre optice.

În continuare, aceste protocoale vor fi comparate din punct de vedere al standardelor, evaluând atât metricile de funcționare, timpul de transmisiune, eficiența codării, complexitatea și consumul mediu de putere. Vor fi scoase în evidență avantajele fiecărui protocol în parte, dar vom pune accent pe utilitatea lor în rețelele de senzori, precum cele utilizate în cadrul acestei lucrări.

2.4.1 Bluetooth și Bluetooth Low Energy

Bluetooth-ul tradițional nu poate fi comparat cu ZigBee sau alte servicii din suita celor prezentate mai sus, întrucât are un defect major, care l-ar descalifica din start din folosirea sa în rețelele wireless de senzori și anume – consumul ridicat de energie. În cazul în care am avea o aplicație practică care ar avea nevoie de alimentare pe bază de baterie, pe o perioadă mai îndelungată de timp, protocolul Bluetooth tradițional (802.15.1) nu ar putea sa fie folosit. Acesta consumă în jur de 1W, valoare care pentru aplicațiile Internet of Things este foarte mare. Bluetooth Low Energy are un consum de aproximativ 10 – 100 ori mai mic, încadrându-se între 10 – 100 mW [8].

Majoritatea caracteristicilor protocolului BLE au fost împrumutate de la Bluetooth, deci se poate discuta în paralel despre acestea, scoțând în evidență doar diferențele majore de interes.

BLE este folosit pentru rețelele personale (PAN), se bazează pe conexiunile wireless pentru scurte distanțe, având un cost scăzut de implementare. În principiu, acest protocol, împreună cu tradiționalul Bluetooth au fost concepute pentru a scădea numărul de cabluri necesare conectării perifericelor la un calculator personal (tastatură, mouse, boxe, cameră web, imprimantă, scanner etc.)

Figura 2.4 – Echipamente interconectate prin Bluetooth [4]

Există două topologii ce sunt folosite în mod special pentru Bluetooth – piconet și scatternet. Piconet are o structură generală ce permite conectarea unui master și a mai multor periferice secundare (slave). Toate perifericele din cadrul acestei topologii se sincronizează între ele cu ajutorul unui clock – rate setat de către master. Pentru a nu exista interferențe, în momentul inițializării conexiunii, masterul este responsabil cu scanarea mediului înconjurător și stabilirea frecvenței de funcționare, astfel încât să nu apară interferențe cu alte conexiuni. Perifericele sunt responsabile de comunicare punct – la – punct doar cu masterul, nu și între ele. Masterul, pe de altă parte, poate avea conexiuni punct – la – punct sau punct – la – multipunct. Pentru economisirea de energie, un periferic (slave), poate intra în mod de stand-by în momentul în care nu are date de transmis.

Scatternet reprezintă o colecție de piconet ce se suprapun în domeniul frecvenței și din punct de vedere spațial. Un dispozitiv ce folosește Bluetooth se poate conecta în mod simultan la mai multe piconet-uri, permițând în felul acesta un flux de date între acestea, devenind astfel, un scatternet.

BLE are avantajul major de a fi suportat de foarte multe sisteme de operare ce se folosesc în prezent pe dispozitivele mobile: Android, iOS, Windows 8/10 sau macOS. Nu multe alte protocoale se bucură de această incorporare, având nevoie de dispozitive suplimentare pentru comunicare activă.

2.4.2 Ultra-Wide-Band (UWB)

Unul dinte protocoale care s-a bucurat de o majoră creștere a popularității în ultima vreme este UWB. Una dintre caracteristicile căreia i se datorează această popularitate este lățimea de bandă posibilă în cadrul transmisiunii de date – aceasta poate să ajungă până la 480 Mbps, fiind mai mult decât suficientă pentru o multitudine de aplicații ce au nevoie de transmisie de date pe rază scurtă, mai ales într-un mediu închis, casnic.

În lipsa interferențelor majore, acesta poate să fie folosit și pe post de înlocuitor al clasicului serial USB 2.0.

Pe lângă acest avantaj se mai adaugă consumul relativ de putere, existând posibilitatea alimentării dispozitivelor cu ajutorul unei baterii, pe o perioadă mai lungă de timp; rezolvarea problemelor de comunicație în cazul apariției mai multor căi de transmisie a datelor și posibilitatea localizării precise a unui nod emițător de date [4].

Totuși, există și un dezavantaj în prezent, la această tehnologie, întrucât durează destul de mult de timp pentru a alege un canal de transmisie. Pe scurt, timpul în care un emițător și un destinatar ajung la sincronizare, este de ordinul milisecundelor. De aceea, este important ca tehnologia de acces la mediul de transmisie (MAC – Medium Access Control) să aibă în vedere acest lucru. Există două protocoale care se ocupă de acest acces – CSMA/CA și TDM.

2.4.3 Wireless Fidelity (Wi - Fi)

În momentul în care se discută despre protocolul Wi-Fi, sunt incluse automat standardele 802.11a/b/g/n/ac/ad, folosite pentru conexiunea într-o rețea locală. Acestea permit conectarea utilizatorului la Internet, când este în apropierea unui AP. Arhitectura generală a protocolului are la bază o celulă ce se numește basic service set (BSS), ce constă într-un set de stații mobile și fixe. Dacă una dintre stații iese din aria de acoperire a BSS-ului la care era conectată, aceasta nu va mai avea conexiune și nu va mai putea comunica direct cu vecinii săi. Mai departe, protocolul IEEE 802.11 a evoluat cu independent basic service set (IBSS) și extended service set (ESS).

Există și posibilitatea interconectării mai multor BSS-uri, cu ajutorul unui sistem distribuit, conducând în felul acesta la apariția unui ESS, cu dimensiuni și complexitate ce pot fi stabilite de către administratorul de rețea.

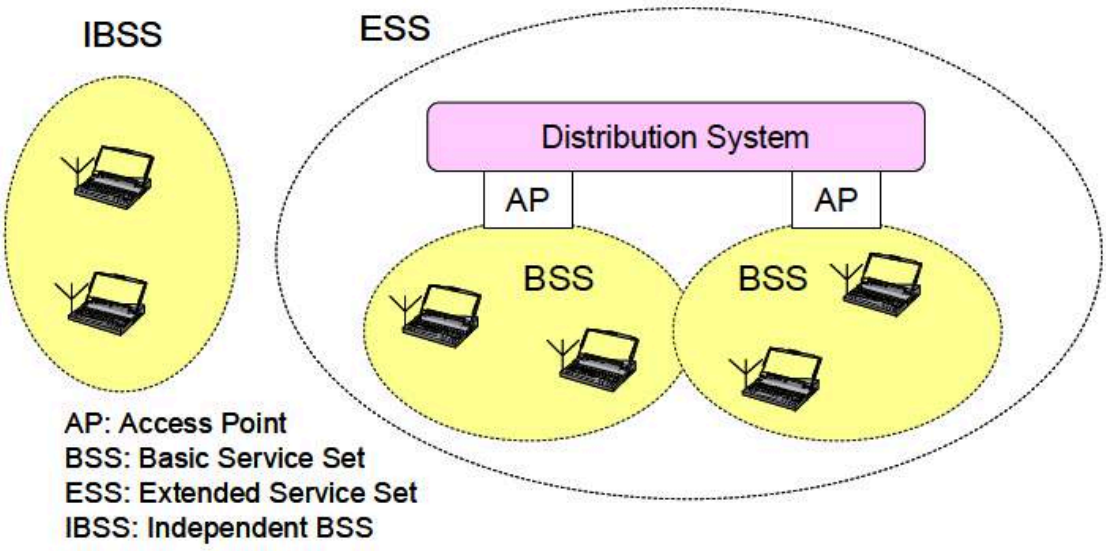


Figura 2.5 – Principiul sistemelor Wi-Fi [4]

Avantajele acestor protocoale sunt viteza foarte mare de transmisie (ultimele versiuni permit traficul de date la o viteza de 1.3 GB/s) și o rezistență ridicată la interferențe cauzate de mediul de transmisiune. Totuși, în cadrul rețelelor wireless de senzori, precum cele ce urmează să fie prezentate, nu viteza este elementul principal ce trebuie urmărit.

Un dezavantaj major este necesitatea echipamentelor intermediare, a căror preț poate crește exponențial în momentul în care se discută conectarea mai multor noduri (peste 10-20). Deși se pot conecta mai multe noduri și la echipamentele ieftine, componentele interne ale acestora (CPU, memorie RAM, ROM), nu fac față volumului de trafic.

2.4.4 ZigBee

ZigBee, așa cum este descris în cadrul standardului IEEE 802.15.4 [7], este un protocol de transmisie de date cu o rată mică de transmisie pentru WPAN. Acesta a fost conceput pentru o conexiune simplă între dispozitive, păstrând consumul de energie la un minim.

Rețeaua cu ZigBee se auto-organizează, fiind necesar un minim de intervenție al utilizatorului sau al administratorului, în momentul configurării inițiale. Intervențiile ulterioare sunt necesare doar în situații cu probleme majore, în care se defectează un număr foarte mare de noduri sau în cazul în care sunt șterse și resetate configurațiile ce rulează. Rețelele organizate de către ZigBee pot fi atât de tip multi-hop, stea (star), cât și de tip mesh (plasă).

Figura 2.6 – Sisteme interconectabile cu ZigBee [13]

În cadrul unei rețele ZigBee, dispozitivele au două moduri în care pot funcționa: dispozitiv complet funcțional (FFD) sau dispozitiv cu funcționare redusă (RFD). Dispozitivele FFD pot îndeplini trei roluri în cadrul unei rețele: coordonator al PAN, coordonator sau periferic. Un dispozitiv FFD poate comunica în activ cu alte FFD-uri sau cu alte RFD-uri, în timp ce un dispozitiv RFD poate comunica doar cu alte dispozitive FFD [11].

Nodurile ce îndeplinesc rolul de dispozitiv cu funcționare redusă au scopuri mai puțin importante în cadrul unei rețele – de cele mai multe ori, au un rol pasiv (întrerupător, senzor pasiv cu infraroșu). Nu au necesitatea de a transmite cantități mari de date și pot comunica cu un singur DCF într-un moment de timp, dar pot face trecerea de la unul la altul.

După ce un nod complet funcțional a fost activat pentru prima oară, este capabil să își formeze propria rețea și să devină coordonator de PAN, formând o rețea stea. Mai multe rețele de tip stea pot opera independent unele de celelalte, separându-se cu ajutorul unui identificator al rețelei, unic în spațiul radio de operare. Odată ce a fost ales un identificator pentru PAN, coordonatorul poate permite conectarea altor noduri în cadrul rețelei.

Datorită avantajelor protocolului (interferențe reduse), posibilitatea criptării datelor, posibilitatea conectării unui număr foarte mare de noduri în cadrul unei singure rețele (> 65000), ZigBee se

pretează foarte bine unor aplicații din domeniul Internet of Things, dar și altora mai complexe. Câteva exemple sunt: automatizarea proceselor casnice și de la locul de muncă (pornirea cafetierei, mașinii de spălat, a frigiderului etc.), monitorizare medicală (EEG, EKG), monitorizare a securității și monitorizare seismologică. În cadrul tuturor acestor aplicații, nodurile compatibile ZigBee pot fi alimentate cu baterie, consumul de energie asociat protocolului fiind foarte redus.

Mai multe detalii despre acest protocol urmează să oferim în următoarele capitole.

În cadrul tabelului 2.1 prezentăm caracteristicile generale ale protocoalelor prezentate, dar și a lui ZigBee, care urmează să fie detaliat în continuare.

Tabel 2.1 – Comparație a protocoalelor wireless[2]

Standard	Bluetooth	UWB	Wi-Fi	ZigBee
IEEE spec.	802.15.1	802.15.3	802.11/a/b/g/n/ac/ad	802.15.4
Banda de frecvențe	2.4 GHz	3.1 – 10.6 GHz	2.4 GHz; 5 GHz	868/915 MHz; 2.4 GHz
Rata transmisie pentru semnal maxim	1 Mb/s	110 Mb/s	Până la 1.3 Gb/s	250 Kb/s
Raza acoperită	10m	10m	100m	10 - 100m
Putere de transmisiune	0 – 10 dBm	-41.3 dBm/Mhz	15 – 20 dBm	(-25) 0 dBm
Banda unui canal	1 MHz	500 MHz – 7.5 GHz	22 MHz	0.3/0.6 MHz; 2 MHz
Celula de bază	Piconet	Piconet	BSS	Stea
Extensia celulei de bază	Scatternet	Punct-la-punct	ESS	Plasă
Numărul maxim de noduri	8	8	>20, 25	>65000
Criptarea datelor	E0	AES1	RC4 (WEP[5])	AES
Autentificare	Cheie secretă	CBC-MAC[2] (CCM)	WPA2[4]	CBC-MAC (CCM)
Protecția datelor	CRC[3] 16-biți	CRC 32-biți	CRC 32-biți	CRC 16-biți

2.4.5 Concluzii

Fiecare protocol în parte se prezintă cu caracteristici și avantaje ce îl recomandă pentru folosirea în diverse aplicații, dar în același timp, și cu dezavantaje pentru aplicațiile propuse în cadrul lucrării prezentate sau în cadrul proiectelor viitoare de același tip.

După cum am ilustrat în cadrul prezentării de mai sus, ne vom axa asupra protocolului ZigBee și vom demonstra utilitatea acestuia în diverse aplicații, cu o complexitate din ce în ce mai mare. Am ales acest protocol deoarece îl consider ca fiind unul dintre cele mai utile și ”de viitor”, întrucât, spre diferență de orice alt protocol, permite conectarea simultană a până la 65.000 noduri, sau chiar și mai mult decât atât.

Pe scurt, el poate să fie folosit la scară mare, având nevoie de o configurare corectă și la început, astfel încât să reziste diverselor mici probleme ce pot interveni, putând ulterior să funcționeze o durată foarte lungă de timp fără intervenția unui administrator.

În principiu, nu am ales folosirea Bluetooth sau Wi-Fi deoarece consumul de energie ridicat ar fi făcut protocolul inadecvat pentru lucrul în cadrul rețelelor de senzori. De asemenea, după cum am prezentat deja în tabel, aceste protocoale nu permit conexiunea între dispozitive pentru prea multe dispozitive. Mai mult decât atât, pentru a permite o conectare de tip point – to – multipoint, dispozitivele trebuie să ruleze un software care consumă mai multe resurse hardware interne, iar consumul acestora crește pe măsură ce crește și numărul de dispozitive conectare. ZigBee permite acest tip de conexiuni în mod implicit, iar consumul de energie și resurse nu crește semnificativ odată cu creșterea numărului de dispozitive conectate.

Un alt motiv care m-a împins să aleg ZigBee ca și principal protocol descris în cadrul acestui lucrări este reprezentat de raza de acțiune a acestuia – cu echipamentul potrivit, ZigBee poate oferi conectivitate eficientă pe o rază de până la 100 metri. Alte protocoale sunt de asemenea capabile să atingă această distanță, dar întrucât mi-am propus să lucrez în mediul ”Internet of Things”, iar scenariile pe care le voi prezenta se desfășoară în principiu într-un mediu casnic sau de birou, nu este neapărată nevoie să ajungem la această distanță – în concluzie, nu am avea de ce să consumăm energie și putere de alimentare din moment ce distanța maximă pe care ne propunem să emitem/recepționăm nu depășește 20 – 25 metri.

CAPITOLUL 3
PROTOCOLUL ZIGBEE

3.1 CARACTERISTICILE PRINCIPALE

În primul rând, după cum este enunțat și la începutul lucrării [2], îmi doresc să stabilesc de la început că există o diferență între ZigBee și XBee, întrucât în cadrul documentării făcute pentru lucrarea curentă, am observat că se creează o confuzie între acești termeni, existând o ambiguitate. ZigBee reprezintă standardul folosit pentru comunicații wireless cu consum redus de energie și putere, în rețele de tip plasă (mesh), în timp ce XBee este un echipament hardware ce suportă o multitudine de protocoale wireless, printre care ZigBee (802.15.4), dar și Wi-Fi.

Pentru a permite comunicare între un număr cât mai mare de dispozitive, fără a ține cont de producător, avem nevoie de o arhitectură de protocol care, la nivelul fizic să ofere a separare a componentelor și funcționalităților în mai multe module, care să poată fi asamblate în diverse metode. Fiecare rețea, indiferent de modul de interconectare a dispozitivelor care o alcătuiesc, are nevoie de un nivel fizic pe care să se transmită semnalele. Rețele de calculatoare se pot interconecta cu ajutorul cablurilor Ethernet, rețelele metropolitane pot fi realizate cu ajutorul cablurilor coaxiale, fibră optică etc., iar exemplele pot continua.

După cum am subliniat însă și mai devreme, tendința generală a prezentului este de a folosi protocoale wireless pentru nivelul fizic – pentru rețelele de calculatoare, protocolul cel mai des folosit este WiFi, în timp ce pentru rețele mobile de comunicații se folosesc protocoale GSM (Global System for Mobile

Communications): 3G, 4G, GPRS, EGRPRS (EDGE), LTE etc. Din punct de vedere al aplicațiilor, al nivelelor superioare din stivele de protocoale (TCP/IP sau ISO/OSI), nu se va simți nicio diferență în momentul în care se face trecerea de la comunicațiile clasice, pe fir, la cele fără fir (cel mult vor apărea delay-uri, dar dacă aplicația este realizată corect, nu ar trebui să facă o diferență majoră). Când vorbim despre nivelul aplicație, ne referim la orice interacțiune directă cu utilizatorul (browser web, jocuri, întrerupătoare, becuri etc.). Aceste aplicații sunt ”protejate” de către interfețele dintre straturile stivelor de protocoale, care permit fiecărui modul software sau hardware să își modifice modul de funcționare, păstrând în continuare comunicarea cu celelalte nivele în exact același fel.

O echivalență, în vederea simplificării percepției și înțelegerii a acestor lucruri se poate face cu ajutorul modului de funcționare a unei mașini. Aceasta poate să meargă peste diverse tipuri de sol: străzi asfaltate, drumuri forestiere, drumuri cu noroi – fără a-și modifica comportamentul. Cauciucurile sunt cele care permit acest lucru, oferind o interfață între nivelul vehiculului (cel al aplicației în cazul nostru) și cel fizic (drumul parcurs). Mașina poate să fie reprezentată ori de vehiculul personal, ori de un camion, motocicletă și așa mai departe – deci aplicația poate să se schimbe, cât timp interfețele sunt prezente.

Nivelul de rețea ce se află sub ZigBee ce suportă o multitudine de caracteristici avansate este cunoscut sub numele de IEEE 802.15.4 – acesta reprezintă un set de standarde ce impun mai multe proprietăți funcționale, cum ar fi: managementul consumului de energie, tipul de adresa, corecția de erori, formatul mesajelor generale, precum și alte specificații tehnice necesare comunicațiilor de tip point – to – point. O parte dintre acestea reprezintă necesități pentru a permite comunicarea între mai multe noduri wireless, în timp ce o parte au fost impuse pentru creșterea performanțelor, în raport cu alte protocoale.

Ca și noduri, în cadrul lucrării prezentate vom lucra cu modulele XBee Series 2 (am fi putut lucra și cu Series 1[6], dar au performanțe mai mici pe termen lung de funcționare) – acestea suportă protocolul 802.15.4 în mod nativ, nefiind nevoie să instalăm componente software pentru interoperabilitate. După cum am spus deja, ZigBee reprezintă un set de nivele construite deasupra standardului 802.15.4 – acestea aduc trei lucruri foarte importante în cadrul unei rețele:

1. Rutare

    - În cazul cel mai general, rutarea reprezintă procesul de găsire al celui mai scurt drum de la o sursă la o destinație, pe baza informațiilor oferite de către rețea. Acest drum este, în general, în cadrul rețelelor, stabilit de către fiecare nod intermediar.
    - Tabelele de rutare definesc modul în care un nod radio wireless poate trimite mai departe (forwarding) mesajele către alte module wireless din apropierea sa, în vederea ajungerii până la o destinație specificată.
    - Tabelele de rutare pot fi populate în mod static sau dinamic, în funcție de alegerea celui care configurează nodurile – dinamic presupune folosirea unui protocol dinamic de comunicare între noduri (pe lângă ZigBee) care să decidă căile cele mai scurte, în timp ce modul static presupune configurarea de către administrator a tuturor căilor.

2. Crearea de rețele ad-hoc

    - Rețelele ad-hoc reprezintă rețele ce se creează în mod automat, fără a fi necesară intervenția unui administrator.
    - Sunt foarte utile în anumite circumstanțe, dar în cazul în care avem o aglomerare de noduri, rețelele se pot crea în mod diferit față de ceea ce mi-aș fi dorit și pot conduce la o funcționare mai puțin eficientă decât cazul ideal.

3. Reconfigurare automată sub formă mesh

- Rețelele de tip mesh reprezintă structuri în care oricare două noduri sunt interconectate prin mai mult de o cale disponibilă, existând oricând o cale redundantă. Căile redundante trebuie să depindă cât mai puțin unele de celelalte.
- Reconfigurarea automată reprezintă capacitatea unei rețele de a-și reveni singură în cazul în care unul sau mai multe noduri nu mai sunt disponibile sau sunt afectate de către un factor extern care le face imposibilă comunicarea. Pe scurt, rutele rămân disponibile, chiar și după ce anumite noduri au devenit indisponibile.

3.2 Rețele formate cu ajutorul ZigBee

O rețea, indiferent de protocolul cu care aceasta este constituită sau de dispozitivele care o constituie, are mai multe tipuri de componente, care îndeplinesc diverse roluri. În cadrul rețelelor în care folosim ZigBee, există trei tipuri de componente, care îndeplinesc funcționalități specifice. Multe rețele pot conține doar două tipuri de componente, altele le vor conține pe toate trei [2]. Decizia legată de acest lucru aparține administratorului, care, pe baza unei analize, hotărăște de ce funcționalități are nevoie și de care se poate folosi. Pentru dezambiguizare, când discutăm despre rețele, ne referim la cel puțin două noduri interconectate – se pot obține diverse funcționalități minimale și cu un singur nod wireless, dar acestea nu vor fi prezentate în cadrul lucrării prezentate.

ZigBee, ca și protocol, a fost proiectat în mod special pentru conceptul de Internet of Things și de automatizări casnice.

3.2.1 Tipurile de dispozitive în cadrul unei rețele

1. Coordonator

- Orice rețea ZigBee are nevoie în orice moment de timp de către un coordonator. Acesta trebuie să fie unic la nivel de rețea.
- Un coordonator este responsabil cu formarea topologiei logice a rețelei, oferire și configurarea adreselor și managementul altor funcționalități ce definesc rețeaua, o securizează și o păstrează funcțională pentru intervale lungi de timp.
- Rolul unui coordonator poate fi asumat de către un alt nod, în cazul în care principalul devine indisponibil.
- Se recomandă ca coordonatorul să aibă o sursă de energie cât mai sigură, pentru a nu pune în pericol funcționarea rețelei. De asemenea, coordonator se alege și nodul care are cele mai bune resurse hardware, pentru a putea face față unui număr mare de conexiuni active în orice moment de timp.

2. Router

- Un router reprezintă un nod wireless ce are acces la toate caracteristicile specifice protocolului ZigBee.
- rețea poate avea mai multe noduri care să îndeplinească funcționalitatea de router.
- Se poate conecta în mod activ la o rețea (nu are deci nevoie de un administrator care să-i spună să se conecteze) sau poate cere să se alăture unei rețele, dacă aceasta este securizată.

- Poate trimite și primi informații de la alte noduri, dar cel mai important – poate ruta informațiile primite. Acest lucru înseamnă că dacă un mesaj a ajuns la router dar nu îi este destinat acestuia, el se poate ocupa cu trimiterea mai departe a informațiilor, către nodul corespunzător, cu ajutorul tabelei de rutare.
- Routerele trebuie de asemenea să aibă o sursă de alimentare constantă, pentru a nu se risca discontinuități în procesul de comunicare.

3. End-device (dispozitiv terminal)

- Aceste dispozitive sunt folosite în principiu, în circumstanțele în care capacitățile hardware și consumul de energie al unui router nu sunt necesare. Ele sunt de fapt versiuni mai reduse ale routerelor.
- End-device-urile pot să se alăture în mod pasiv rețelelor din jurul său și sunt capabile să trimită și să primească informații, dar astea sunt cam toate capacitățile sale.
- Un end-device are nevoie în orice moment de timp de un router sau un coordonator la care să se conecteze, pentru a reprezenta ”părintele”. Un end-device nu se poate conecta în mod activ într-o rețea și depinde de existența unui ”părinte” – acesta îi ajută să se conecteze la rețea și de asemenea stochează mesaje când end–device-urile sunt în stand-by.
- rețea poate conține un număr foarte mare de end-device-uri – de fapt, bazându-ne pe caracteristicile ZigBee, putem forma o rețea care să conțină doar un coordonator și multe end-device-uri, fără niciun router. Totuși în aceste circumstanțe, performanțele vor fi limitate
- La un end-device, de cele mai multe ori, se vor conecta senzori – direct sau indirect cu ajutorul unei plăci de dezvoltare (cum ar fi Arduino)

3.2.2 Topologiile disponibile

O topologie reprezintă modul de interconectare al dispozitivelor în cadrul unei rețele – modul în care se coordonează și își transmit informațiile între ele. O topologie poate să fie logică sau fizică – amândouă sunt la fel de importante. Pentru a concepe o topologie completă și corectă este necesară o analiză cuprinzătoare asupra a ceea ce se dorește implementat, luându-se în considerare o multitudine de factori, printre care:

- ✓ Numărul de dispozitive conectate în orice moment de timp (în cazul nostru, se discută despre noduri);
- ✓ Mobilitatea nodurilor conectate – în cazul rețelelor de senzori wireless, ne va interesa raza de acțiune a protocolului, întrucât o parte dintre senzori vor putea fi mobili;
- ✓ Lățimea de bandă necesară fiecărui nod – majoritatea scenariilor pe care urmează să le ilustrăm nu necesită o lățime foarte mare de bandă, întrucât volumul de date transmis într-o secundă nu este foarte mare;

Topologia fizică reprezintă poziționare efectivă într-un mediu a nodurilor, conținând distanța dintre acestea și mediul folosit pentru interconectare (în cazul nostru vom avea numai mediul wireless). Topologia logică conține informații legate de adresarea nodurilor, protocoalele folosite, securitatea impusă în cadrul rețelei, rolul jucat de fiecare nod în parte etc. Cele două topologii diferă în majoritatea cazurilor.

În cadrul rețelelor constituite cu ajutorul ZigBee, avem la dispoziție, în principiu, patru tipuri de topologii logice [5], pe care urmează să le descriem în continuare. Topologiile fizice nu au o relevanță foarte mare în cadrul acestor rețele, întrucât mediul de transmisiune va fi întotdeauna reprezentat de către aer, iar poziționarea va fi relativă la locația în care se dorește implementarea unui exemplu/scenariu.

1. Pereche (Pair)

Reprezintă cea mai simplă modalitate de interconectare a oricare două noduri/module wireless. După cum am precizat și mai sus, orice rețea are nevoie de un coordonator unic – așadar, în cadrul topologiei, unul dintre noduri trebuie să fie reprezentat de acesta, în timp ce celălalt va fi reprezentat de către un router sau un end-device.

Această topologie este ideală pentru scenariile simple și pentru detectarea, și rezolvarea problemelor dintr-o topologie mai mare – se izolează pe rând nodurile, iar cu ajutorul unui nod de test, de a cărui funcționalitate corectă suntem siguri, se testează fiecare dintre acestea, în rețele de tip pereche.

Figura 3.1 – Topologie de tip pereche (pair) [2]

2. Stea (Star)

Acest tip de rețea este de asemenea, în principiu, destul de simplu de înțeles și implementat – un nod ce îndeplinește funcția de coordonator se află în centrul topologiei, iar el se conectează, la un număr de end device-uri. Orice mesaj trimis în cadrul acestui tip de rețea trebuie să treacă, în mod obligatoriu, prin coordonator, care se ocupă cu rutarea lor în mod corespunzător, către destinația finală.

End-device-urile nu au comunicare directă între ele în cadrul acestui tip de topologie – avem astfel avantajul că nu pot apărea suprapuneri de date, reducând posibilitatea pierderii informațiilor furnizate de către orice end-device la orice moment de timp.

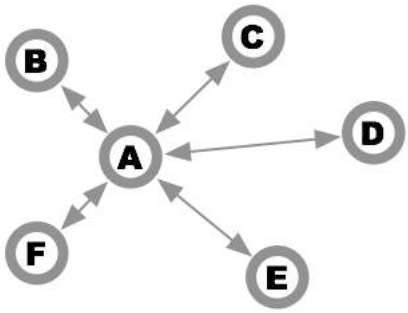


Figura 3.2 – Topologie de tip stea (star) [2]

3. Plasă (Mesh)

Complexitatea rețelelor crește în momentul în care se folosește acest tip de topologie, întrucât avem prezente în mod concomitent toate cele trei tipuri de componente – coordonator, router și end-device. Routerele sunt responsabile cu trimiterea mesajelor către alte routere sau către end-device-uri atunci când este necesar. Coordonatorul (reprezintă mai mult o formă avansată de router) este responsabil cu managementul rețelei și poate de asemenea să ruteze mesaje. End device-urile se pot lega atât de coordonator, cât și de router – ele pot genera și primi trafic de date, dar vor avea nevoie întotdeauna de un ”părinte” care să le ajute să comunice cu alte noduri ale rețelei.

Rețelele ce sunt construite pe baza acestei topologii se diferențiază față de cluster-tree prin faptul că au un grad ridicat de redundanță – asta înseamnă că un router se conectează simultan la coordonator dar și la alte routere – dacă apare o problemă cu o cale de comunicare, există întotdeauna una redundantă. În majoritatea circumstanțelor acest lucru reprezintă un avantaj, dar poate fi considerat un dezavantaj când este necesară calcularea constantă a celor mai scurte drumuri, care să fie instalate în cadrul unei tabele de rutare.

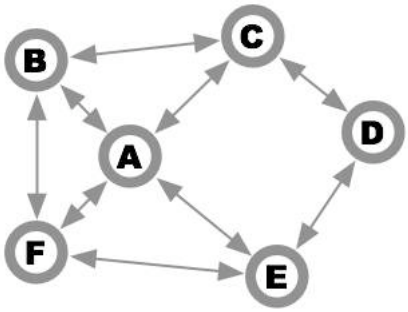


Figura 3.3 – Topologie de tip plasă (mesh) [2]

4. Arbore (Cluster-tree)

Se asemănă destul de mult cu topologia mesh, dar, în acest tip de topologie, routerele formează o formă de ”back bone” a rețelei, care se conectează doar la coordonator. Nu mai există niciun fel de redundanță a rețelei, iar defecțiunea unui router, aduce o dată cu sine, imposibilitatea de comunicare a anumitor end-device-uri.

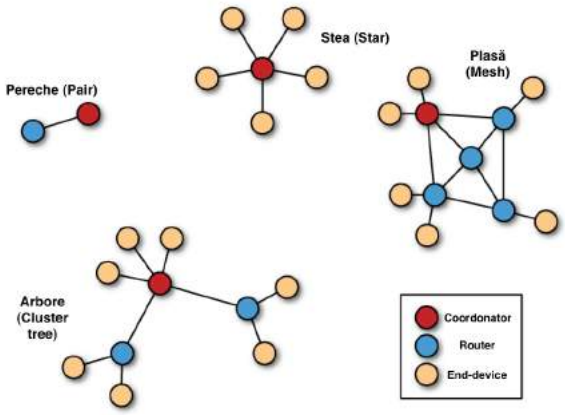


Figura 3.4 – Topologii disponibile în ZigBee

3.2.3 Adresarea nodurilor

Pentru a fi posibilă comunicarea între oricare două dispozitive, în cazul general, este necesar ca cele două să aibă stabilite adrese, de același fel, cu ajutorul cărora să se identifice destinația intenționată. În cadrul protocolului ZigBee se respectă această regulă, niciun mesaj neputând fiind transmis de către un nod fără o adresă destinație stabilită.

Într-un mod asemănător adresării folosite în rețelele Ethernet – IPv4, ZigBee folosește două tipuri de adrese pentru identificarea sursei și a destinației. În primul rând, fiecare nod ZigBee este identificat, la nivel global cu o adresă serială pe 64 de biți, reprezentată în format hexazecimal. Aceste adrese sunt unice, neexistând, teoretic, două noduri fabricate cu aceeași adresă. Pentru a putea fi ușor de citit și identificat, aceste adrese sunt inscripționate pe spatele nodurilor. În figura 3.5 se poate observa adresa unuia dintre nodurile folosite în cadrul acestei lucrări (un nod de tip XBee, care va fi detaliat în capitolele următoare).

Figura 3.5 – Exemplu adresă nod XBee

Pe lângă această adresă, se mai folosește o adresă mai scurtă, pe 16 biți, alocată în mod dinamic de către nodul coordonator în cadrul unei rețele funcționale. Această adresă nu mai este unică decât la nivelul unei rețele wireless de noduri, motivul principal pentru care este folosită fiind reprezentat de către dimensiunea mai redusă a pachetelor în cadrul transmisiei și de către memoria relativ limitată a echipamentelor.

În cele din urmă, un nod XBee poate să primească și o adresă reprezentată sub forma unui șir de caractere, cu ajutorul căruia să fie mai ușor unui administrator să urmărească configurarea și comportamentele membrilor unei rețele. Această adresă nu este folosită în cadrul transmisiunii de date, ci doar pentru monitorizare și verificare, fiind utilă minții umane - într-un mod asemănător sunt folosite numele pentru identificarea site-urilor și nu adresele IP publice, unice la nivel global.

Tabel 3.1 – Adresare nodurilor XBee

Tipul de adresă	Exemplu	Unicitate
64-biți	0013A20040ABC313	Da, la nivel global – alocat de către producător
16-biți	7FB8	Da, la nivel de rețea – alocat de către coordonatorul rețelei
Șir de caractere	Horia's XBee	În funcție de decizia administratorului rețelei de noduri

În continuare, fiecare rețea în parte va avea un identificator propriu, pentru a asigura sincronizarea dintre nodurile pe care le doresc să comunice între ele – în felul acesta, protocolul ne permite să avem mai multe noduri în aceeași locație, dar care să facă parte din rețele diferite – cu o sincronizare corectă, nu vor avea loc nici interferențe (despre canalele folosite de către ZigBee discutăm în capitolele următoare). Rețelele create cu ajutorul protocolului ZigBee se încadrează în categoria PAN (Personal Area Network – Rețele personale), iar identificatorul folosit pentru acestea este de asemenea pe 16 biți, reprezentat sub forma hexazecimală.

Putem concluziona, după analiza adresării ce ne stă la dispoziție, că avem posibilitatea definirii unui număr de 65.536 (2^{16}) rețele unice la nivelul unei suprafețe clar delimitate, fără a exista riscul suprapunerii între două dintre acestea. În cadrul fiecărei rețele, avem posibilitatea alocării adreselor unice către 65.536 de noduri – în felul acesta putem spune că într-o suprafață în care funcționează protocolul ZigBee putem avea peste 4 miliarde de dispozitive (4.294.967.296) care să comunice simultan pentru transmisia de date. Această valoare este echivalentă cu numărul teoretic de adrese unice IPv4 ce pot fi alocate la nivel global în Internet.

Trebuie subliniat că la această valoare se ajunge doar dacă este optimizat și procesul de alocare al canalelor și nu există interferențe între noduri foarte apropiate – în cazul în care nu este respectat acest lucru, eficiența transmisiei va scădea, îngreunând comunicarea.

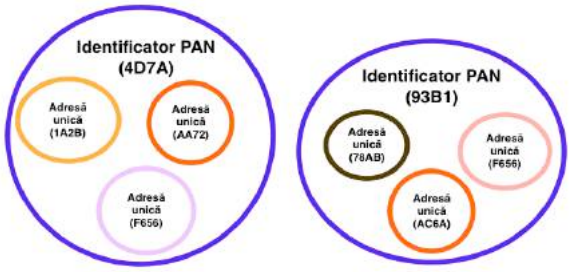


Figura 3.6 – Identificatori într-un PAN ZigBee

3.2.4 Canalele de comunicație

Pe lângă responsabilitatea alegerii unui identificator pentru PAN și alocarea adreselor unice la nivel de rețea, coordonatorul mai trebuie să aleagă și un canal pe care să se sincronizeze și să comunice cu nodurile din rețea. Canalul trebuie să fie unic pentru o rețea, iar în majoritatea cazurilor, coordonatorul face alegerea după ce le scanează pe toate cele disponibile și îl identifică pe cel mai puțin aglomerat.

Spre diferență de alte protocoale, coordonatorul dintr-o rețea ZigBee își poate schimba în mod automat canalul când simte o aglomerare de trafic de date, semnalând acest lucru către routere și end device-uri.

Standardul IEEE 802.15.4 oferă trei moduri de operabilitate, în următoarele benzi de frecvență: 2.4GHz (disponibil la nivel mondial), 915MHz (Statele Unite ale Americii), 868MHz (Europa). Împărțirea în canale între acestea se face în felul următor:

- Un canal între 868 – 868.6 MHz (20kbit/s)
- 10 canale între 902 – 928 MHz (40kbit/s)
- 16 canale între 2.4 și 2.4835 GHz (250kbit/s)

Toate aceste benzi de frecvență funcționează bazându-se pe tehnica spectrului direct împrăștiat (DSSS). Pentru modulare, banda de 2450 MHz folosește modularea în fază cu offset în cuadratură (OPQSK), în timp ce benzile de 868 – 915 MHz se bazează pe modulare binară în fază (BPSK).

Tabel 3.2 – Caracteristicile canalelor folosite în ZigBee

Proprietăți	Lățimea de bandă	
	2.4 GHz	868 MHz
Rata de bit pentru transmisie de date	250 kbps	20 kbps
Aria de acoperire	În cadrul clădirilor – 30 metri; în exterior – 100 metri	În cadrul clădirilor – 15 metri; în exterior – 75 metri
Latență	15 ms	
Număr de canale	16	1
Numerotarea canalelor	11 - 26	27
Metoda de acces la canal	CSMA – CA	
Modularea	OPQSK (Offest Quadrature Phase Shift Keying)	BPSK (Binary Phase Shift Keying)

În majoritatea cazurilor se va opta pentru folosirea lățimii de bandă de 2.4 GHz, datorită ratei de bit mult mai ridicată (de peste 10 ori, conform tabelului de mai sus) și datorită ariei de acoperire mult mai bună a semnalului. Totuși, odată cu această alegere ne punem problema suprapunerii cu protocolul 802.11 (Wi-Fi), întrucât majoritatea AP-urilor folosesc aceeași frecvență pentru comunicare (chiar dacă în ultimele versiuni, 802.11 permite comunicarea și pe 5GHz și 50GHz – acestea consumă mult mai multă energie).

Această problemă este una reală, întrucât Wi-Fi este principalul protocol folosit în rețelele wireless de transmisii de date, ca alternativă la clasicele rețele Ethernet și cabluri de cupru (UTP – Unshielded Twisted Pair sau coaxiale), fiind practic prezent în aproape orice clădire sau locuință. Wi-Fi și ZigBee

folosesc aceeași lățime de bandă, dar scopul pentru care sunt folosite este radical diferit, ZigBee fiind folosit pentru rețele de senzori, procese de automatizare, monitorizare etc.

După cum se poate observa și în figura 3.7, singurele canale (cu lățime de 22 MHz) ale protocolului 802.11 care nu se suprapun sunt 1, 7, 13 (respectiv 1, 6, 11 pentru SUA). Atât canalul 1 cât și canalul 6 (respectiv 7) se suprapun cu mai multe canale ale lui ZigBee (lățime mult mai mică, doar 2 MHz). Singurul canal al Wi-Fi care nu se suprapune complet cu ZigBee este canalul 11 (respectiv 13).

Din punct de vedere al planificării unei rețele, acest amănunt poate complica puțin lucrurile, dar în realitate nu este atât de dificil să se renunțe în cadrul configurării Wi-Fi la canalul 11, pentru a se permite existența unor canale libere de ZigBee. Iar în eventualitatea în care acest lucru nu este posibil, cele două pot coexista fără prea mari probleme de comunicare – apar delay-uri și pierderi de date semnificative doar în momentul în care ambele rețele au trafic constant și conectate cel puțin 30 – 35 de dispozitive/noduri.

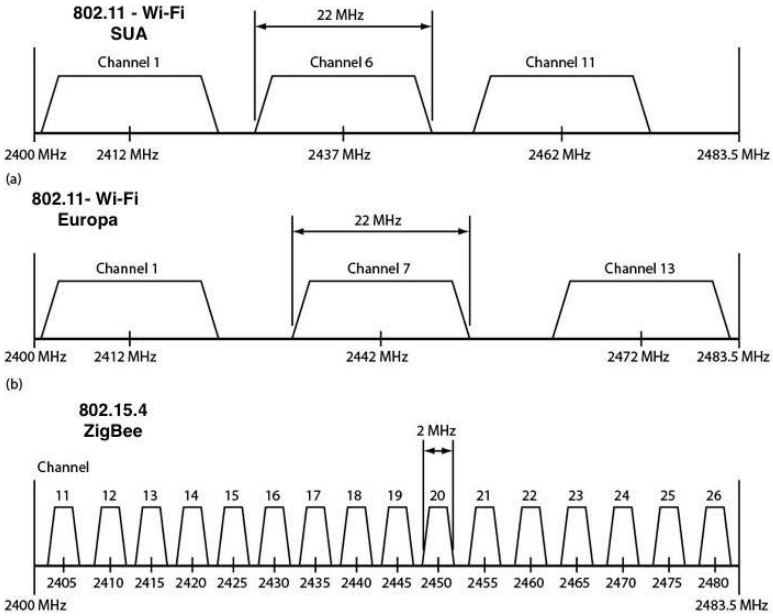


Figura 3.7 – Canelele disponibile în ZigBee și Wi-Fi [6]

3.3 Componentele Hardware Ale Transmisiunii De Date

În capitolele precedente m-am ocupat cu prezentarea generală a protocolului ZigBee și a noțiunilor fundamentale legate de funcționarea sa într-un mediu standard de comunicație. În continuare ne propunem să detaliem modul de transmisiune a datelor, informațiile hardware legate de dispozitive folosite pentru comunicarea cu acest protocol, stările de funcționare a acestor noduri și particularitățile comunicațiilor de acest fel.

3.3.1 Modulele XBee

După cum am precizat anterior la începutul lucrării, există o diferență majoră între ZigBee și XBee – primul fiind protocolul folosit în această lucrare, în timp ce XBee este denumirea modulelor hardware ce sunt compatibile cu acest protocol și sunt folosite în cadrul rețelelor de senzori pentru transmisia și recepția datelor relevante.

Modulele XBee au un design atât hardware cât și software conturat de către Digi International, fiind livrate împreună cu o varietate de antene (cu ajutorul cărora se poate regla puterea de transmisiune și distanța acoperită de semnalul transmis) și module firmware ce pot fi actualizate oricând cu ajutorul unui software. În continuare vom detalia cele mai populare module XBee folosite în prezent, axându-ne pe cele folosite în cadrul lucrării. Există două versiuni de bază a modulelor XBee:

XBee Series 1 – Acestea sunt produse de către Freescale și au rolul de a oferi un mod simplu de transmisiune, bazându-se în principiu pe comunicațiile de tip punct – la – punct. Pe lângă acesta mai este posibilă folosirea unor rețele de tip mesh personalizate acestor module. Nu vom folosi aceste module în cadrul lucrării întrucât ne vom axa pe rețele a căror dimensiune poate fi crescută fără nicio problemă.

XBee Series 2 – Cea de-a doua serie de module XBee sunt dotate cu un microcontroler de la Ember Networks, care le permite funcționarea într-o gamă mai largă de rețele. Rețele de tip mesh sunt optimizate cu acest microcontroler, pe conceptul acestora fiind construite majoritatea rețelelor de senzori. În cadrul lucrării, toate noțiunile prezentate se vor baza pe acest tip de module.

Ambele tipuri de module se găsesc atât în versiuni standard, cât și în versiuni mai profesioniste, diferența fiind dată de către puterea de transmisie a semnalului – versiunile profesioniste se identifică prin cuvântul Pro atașat denumirii – XBee Series 2 Pro. O diferență relevantă între cele două mai este dată de cost, versiunile Pro fiind mai scumpe, datorită specificațiilor tehnice superioare. În cadrul lucrării nu vom folosi modulele Pro întrucât nu avem nevoie de putere foarte mare de transmisie.

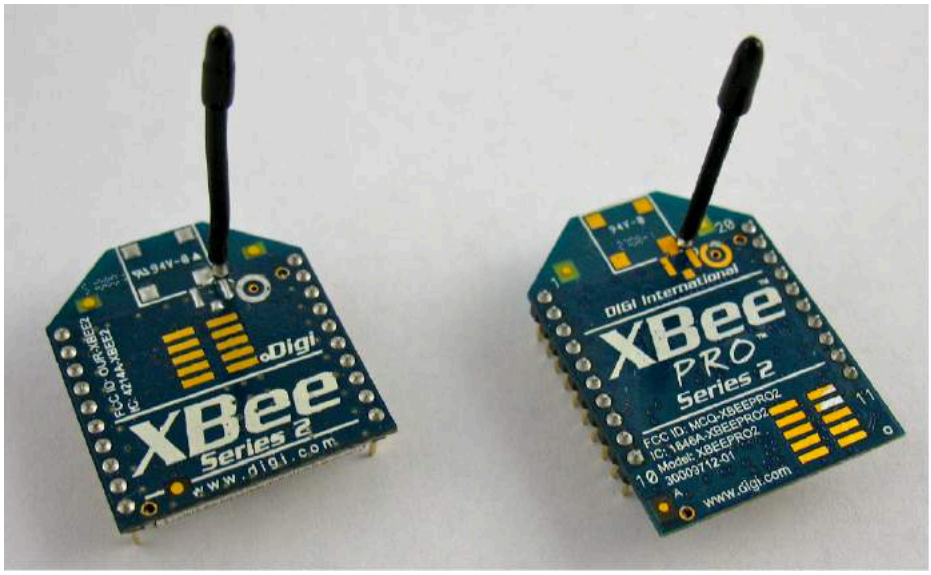


Figura 3.8 – Antene conectate pe modulele XBee [2]

3.3.2 Specificații Tehnice

În timp ce modulele Series 1 sunt foarte populare la început, pentru învățat și proiecte de dimensiuni reduse, modulele Series 2 sunt recomandate datorită faptului că suportă toate proprietățile protocolului ZigBee. Acestea au un design special pentru rețelele de senzori de dimensiuni ridicate, fiind potrivite pentru mediile mari de lucru, comerciale, academice, dar și cele casnice.

Acestea din urmă au o rază mai mare de acoperire, folosind și mai puțina putere pentru alimentare decât Series 1. Formatul fizic este păstrat pentru a putea fi posibilă actualizarea, într-un mod cât mai simplu din punct de vedere hardware, de la Series 1 la Series 2. Totuși, nu există posibilități de interoperabilitate între cele două versiuni – acest lucru înseamnă că dacă se decide migrarea unei rețele de la Series 1 la 2, toate modulele trebuie schimbate, nu doar o parte, întrucât o singură rețea nu poate fi constituită din ambele variante, simultan.

În cadrul tabelului 3.2 prezentăm specificațiile tehnice ale celor două tipuri de module, scoțând în evidență superioritatea modulelor Series 2.

Tabel 3.2 – Comparație între generațiile modulelor XBee

Proprietate tehnică	XBee Series 1	XBee Series 2
Rază de acoperire interior (clădire)	30 de metri	40 de metri
Rază de acoperire exterior (câmp liber)	100 de metri	120 de metri
Curent transmisie/recepție	45/50mA	40/40mA
Firmware preinstalat	802.15.4	ZB ZigBee mesh
Pini digitali intrare/ieșire	8 (doar o intrare)	11
Pini analogici intrare	7	4
Pini analogici ieșire (PWM)	2	0
Interoperabilitate routare rețele mesh, reconfigurare automată, rețele ad-hoc	Da	Da
Rețele punct-la-punct, stea	Da	Da
Rețele mesh, arbore	Nu	Da
Versiune unică de firmware acceptată în rețea	Da	Nu
Necesitatea unui nod coordonator	Nu	Da
Firmware disponibil	802.15.4 (standard IEEE)	ZB (ZigBee 2007), ZNet

Modulele XBee, ca orice alte dispozitive ce funcționează cu unde radio, au nevoie de antene pentru a putea transmite și primi semnale. Există mai multe moduri de a integra antenele pe componentele hardware prezentate, fiecare având propriile avantaje și dezavantaje. Vom prezenta doar câteva în continuare, întrucât în cadrul lucrării nu m-am folosit de antene, nefiind nevoie de ele pentru conceptele demonstrate.

- Antenă cu fir – o simplă legătură conectată la modulul radio – în majoritatea cazurilor este exact ceea ce trebuie pentru comunicații, fiind simplă cu radiații omnidirecționale (puterea semnalului este aproximativ aceeași în toate direcțiile, în linie dreaptă și perpendiculară pe modul)
- Antenă de tip chip – Un modul sub forma unui chip ceramic, atașat direct pe modulul XBee – are avantajul faptului că semnalul nu este propagat în toate direcțiile. Reprezintă alegerea ideală pentru produsele ce se doresc a fi purtate de către utilizator.
- Conector U.FL – Există două tipuri de conectori pentru antenele externe, iar acesta este cel de dimensiuni reduse. Aceste antene nu sunt întotdeauna necesare, fiind folosite doar în momentele în care modulele radio sunt încapsulate (închise într-o cutie), iar antena trebuie să fie în exterior – în felul acesta semnalul nu este atenuat în mod semnificativ.

- Conector RPSMA – Acest conector reprezintă doar o formă mai mare a conectorului U.FL (care este mai fragil). Folosit în mediile în care antena ar putea să fie afectată de șocuri mecanice – avem o siguranță mai mare a faptului că aceasta va rămâne conectată la modulul XBee.

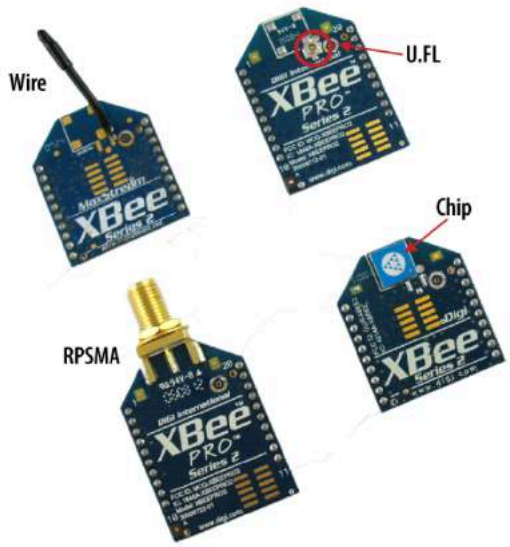


Figura 3.9 – Tipuri de antene disponibile pentru modulele XBee [4]

3.3.3 Conexiunea la Modulele XBee

Vom discuta în continuare despre modul în care utilizatorul uman poate interacționa cu modurile XBee, dar și despre modurile în care acestea pot comunica între ele. Pentru început însă, precizăm că în vederea configurării și comunicării dintre calculator și modulul XBee este necesar un modul specific.

Mai exact, este necesar un adaptor care să permită scrierea și citirea informațiilor de pe modulele XBee pentru utilizatorul uman, împreună cu un software specific ce va rula pe calculatorul de pe care se realizează modificările. Aceste adaptoare se pot conecta direct pe o conexiune serială (USB), având 5 pini principali ce pot fi folosiți (Reset, RX, TX, Masă, alimentare 5V). Modulele folosite în cadrul acestei lucrări nu pot fi utilizate pentru alte scopuri pe lângă comunicarea cu calculatorul, întrucât nu oferă conectivitate către ceilalți pini ai modulului XBee.

Totuși, aceste adaptoare pot fi folosite pentru o conexiune minimală între noduri, în vederea verificării sincronizării unei rețele și a transmisiei de date foarte simplă (cum ar fi un chat).

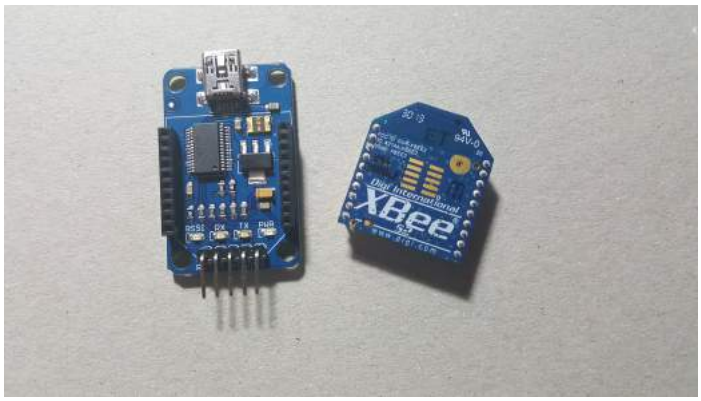


Figura 3.10 – Adaptor pentru configurarea modulelor XBee

Pentru partea de software, există o soluție oferită de producătorii modulelor XBee ce poate fi folosită pentru update-ul firmware-ului, schimbarea rolurilor și modificări de bază legate de funcționalitate. Numele programului este XCTU și poate fi descărcat direct de pe pagina celor de la Digi Communications [14].

În cadrul acestui program este posibilă descoperirea în mod automat a modulelor XBee conectate, împreună cu ”citirea” informațiilor legate de funcționalitatea la un moment dat a acestora, după cum este ilustrat în următoarele figuri:

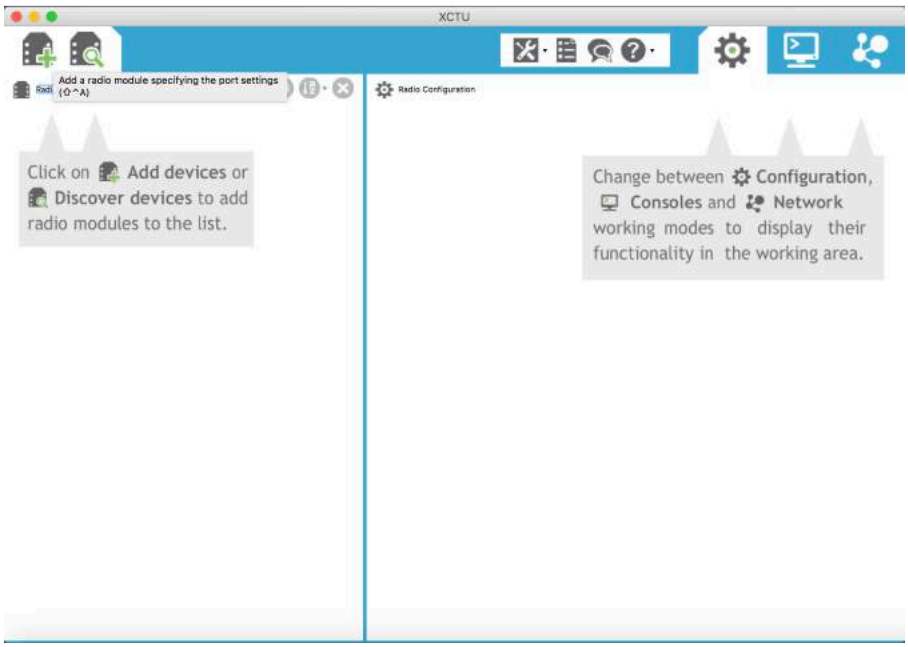


Figura 3.11 – Interfața cu utilizatorul în programul X-CTU

Din interfața ce se deschide în mod automat la pornirea programului selectăm opțiunea de descoperire a modulelor radio conectate, după ce acestea au fost conectate pe legătura serială disponibilă – în cazul de față, USB.

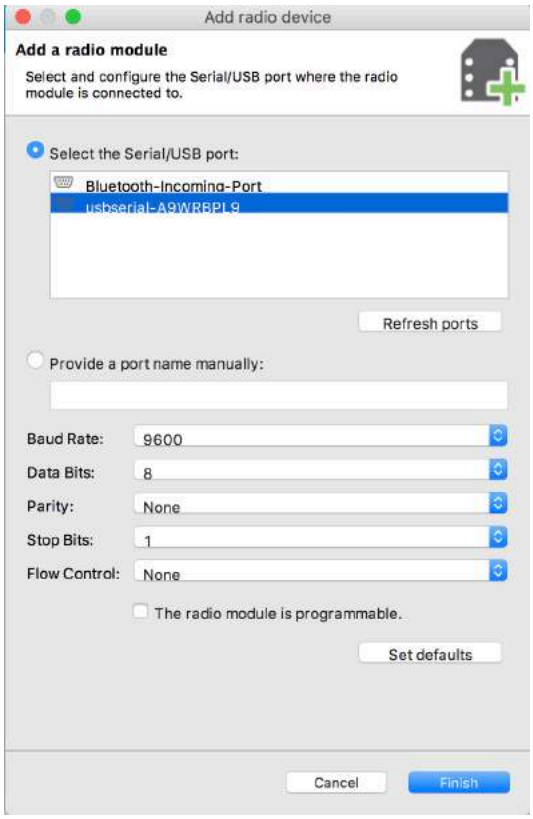


Figura 3.12 – Selectarea portului pentru configurarea modului XBee

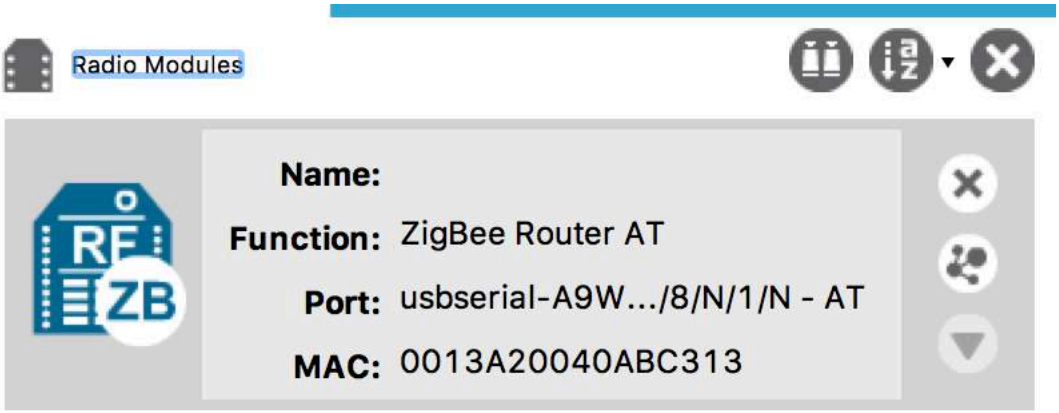


Figura 3.13 – Identificatorii modulului XBee

După finalizarea căutării, modulele radio descoperite vor fi prezentate în cadrul ferestrei principale a aplicației, fiind specificat rolul acestora, portul pe care se află conectate și adresa MAC (adresa fizică) corespunzătoare.

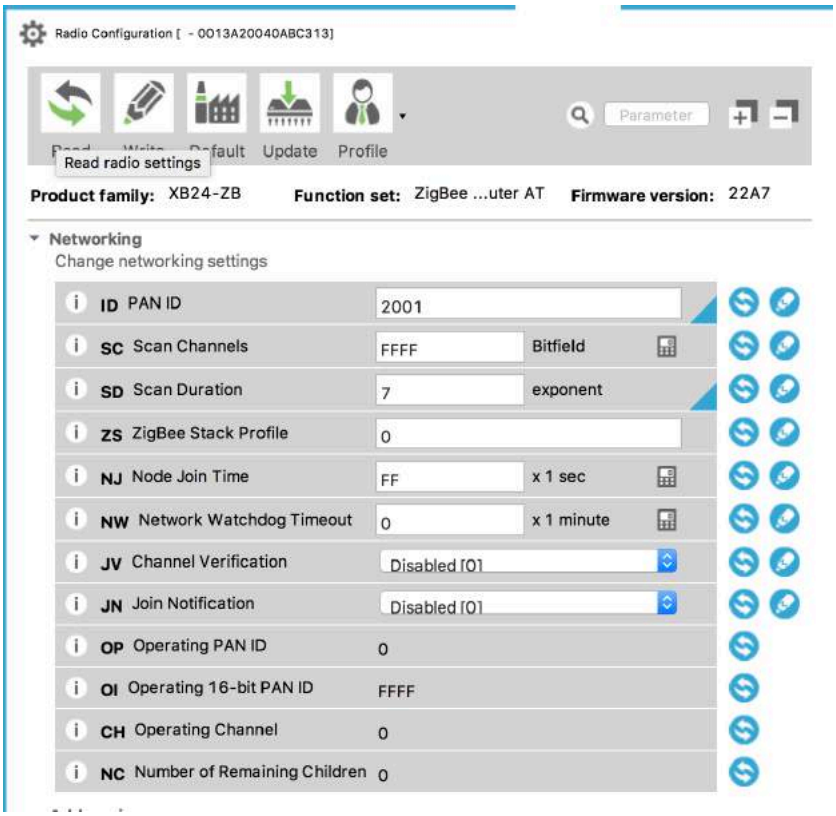


Figura 3.14 – Informații de rețea a modulului XBee

În continuarea selectării unuia dintre modulele descoperite, ne vor fi prezentate toate informațiile configurate până la momentul respectiv și de asemenea, vom avea posibilitatea modificării acestor date cu ușurință (lucru ce ne va fi util în continuare).

Această citire și configurare este realizată într-un mod asemănător celui în care modulele XBee își transmit informațiile între ele. Totuși, în momentul în care se realizează conexiunea serială, datele pot fi încapsulate în două moduri diferite [13]:

- Echipamentele configurate în mod API folosesc o încapsulare a datelor ce ușurează comunicațiile dintre calculatoare, dar este mai dificil de interpretat de către utilizatorii umani
- Modul AT – oferă o serie de comenzi ce sunt concepute special pentru a ușura configurarea de către utilizatorul uman. În cadrul acestui mod de configurare/comunicare, modulele radio se pot afla în două stări [2]:

1. Modul transparent

   Acesta este modul implicit pentru modulele radio ce funcționează în modul AT. Poartă această denumire întrucât radiourile trimit pur și simplu mai departe informațiile primite, fără a le altera sau modifica în vreun fel. Când primește date de la un alt nod, acestea sunt trimise pe legătura serială în mod identic celui în care au fost primite.

2. Modul de comandă

   Acest mod este folosit în momentul în care ne dorim ca nodul să nu trimită mai departe informațiile primite de la calculator, ci mai degrabă să le păstreze în vederea configurării – mai exact, îl folosim când dorim să schimbăm ceva anume în funcționarea nodurilor.

Trecerea între modul transparent și modul de comandă este realizată prin folosirea unei secvențe prestabilite – ”+++”. În momentul în care este primită această secvență, nodul intră în modul de comandă, semnalează acest lucru prin returnarea unui mesaj ”OK”, unde așteaptă comenzi de configurare. Dacă nu se detectează niciun fel de comenzi de configurare timp de 10 secunde, atunci se face în mod automat trecerea în modul transparent.

Pentru această configurare, poate fi folosit meniul de terminal al aplicației XCTU sau un alt program ce emulează o consolă și citește/interpretează informații de pe interfața serială, cum ar fi PuTTy, CoolTerm etc. În imaginea de mai jos este demonstrată trecerea din modul transparent în cel de comandă, precum și modificarea, verificarea și salvarea unui parametru (PAN ID).

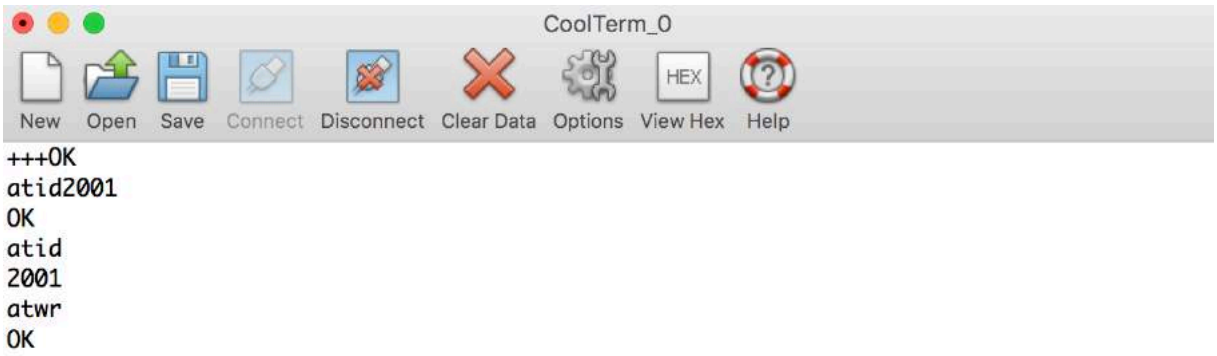


Figura 3.15 – Comenzi de configurare trimise din terminal

3.4 Transmisia De Date Între Modulele XBee

Configurate în modul AT, nodurile XBee au o funcționare destul de simplă de înțeles și utilizat (după cum este ilustrat în capitolul anterior), transmisia făcându-se în mod serial, nefiind necesară segmentare de date sau o sincronizare a acestora. Singura necesitate în cadrul unei rețele formată din module XBee configurate în mod AT de funcționare este ca acestea să formeze o rețea – adică să aibă același PAN ID, unul dintre dispozitive să aibă rolul de coordonator și să funcționeze pe același canal de transmisiune.

În cadrul lucrării prezentate vom folosi modul AT de configurare în cadrul primelor proiecte prezentate pentru a ilustra funcționalitatea de bază a protocolului, urmând ca ulterior, coordonatorul rețelei – cel care practic stochează și afișează datele recepționate, să lucreze în mod API.

API – reprezintă în general un set de interfețe standard de comunicare, concepute special pentru a se permite comunicarea între două programe software, fără a se ține cont de limbajul în care acestea sunt scrise, locația în care acestea rulează, sisteme de operare etc. Pe scurt, cu ajutorul unui API, o aplicație de tip software poate să trimită cereri, către un alt program, formatarea cererii făcându-se într-un standard cunoscut și utilizat de către toată lumea.

În modul AT, aveam nevoie de un terminal pentru a realiza comunicarea cu nodurile XBee, iar informația era afișată doar în acele ferestre. Acest mod este util și pentru comunicarea cu modulele externe, cum ar fi plăcuțele de dezvoltare Arduino. Vom folosi modul API de comunicare pentru a putea permite și altor aplicații să primească informații direct de la modulele XBee pentru a le putea

prelucra și lua decizii pe baza acestora. Bibliotecile necesare comunicării dintre aplicațiile scrise în aproape orice limbaj de programare pot fi integrate cu ușurință în cadrul unui IDE.

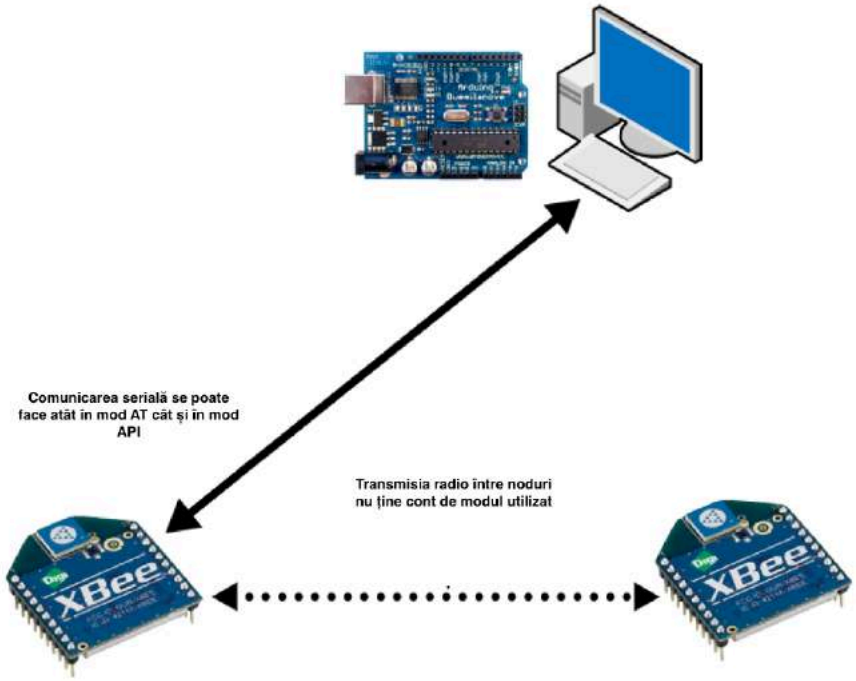


Figura 3.16 – Tipuri de comunicare între nodurile XBee

Specificăm ca nodurile XBee nu au suficientă memorie integrată pe placă pentru a putea reține atât comenzile de configurare pentru modul AT cât și cele specifice modului API. În consecință. pentru a putea face trecerea dintr-un mod în altul, este necesară încărcarea unui firmware diferit, lucru care se poate realiza cu ajutorul adaptoarelor prezentate mai devreme și a programului XCTU.

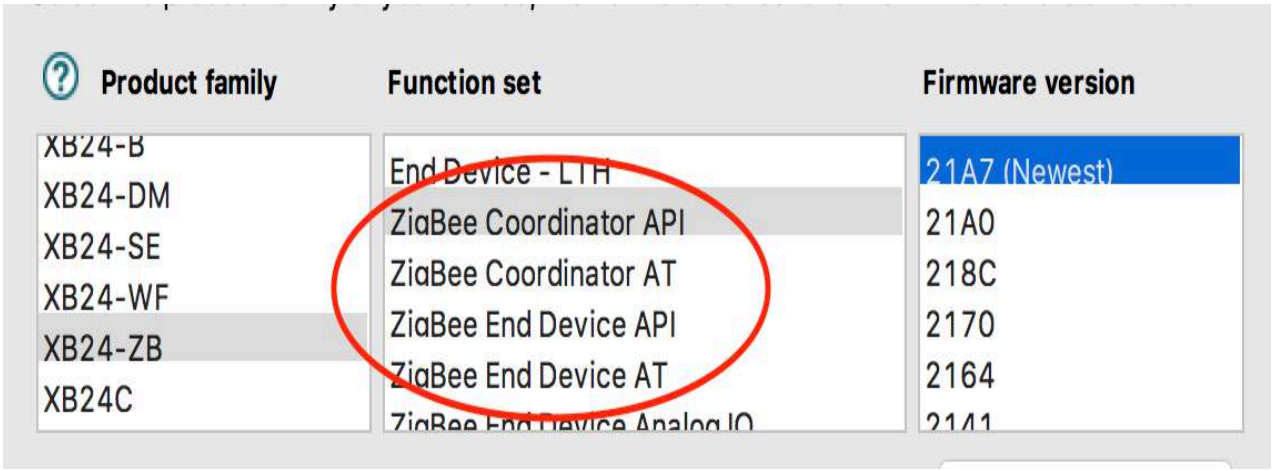


Figura 3.17 – Rolurile XBee în cadrul unei rețele

Scopul modului API este acela de a oferi o transmisie de date predictibilă, sigură și eficientă, prin intermediul bibliotecilor software corespunzătoare. În continuare prezentăm structura cadrelor folosite pentru asigurare sincronizării între programele software ce comunică între ele și transmisia efectivă de informații.

Tabel 3.3 – Structură general a unui cadru de date

Delimitator start (preambul)	Lungime cadru	Date (payload)	Checksum (verificator)
Byte 1	Byte 2 - Byte 3	Byte 4 ... Byte N	Byte N+1
0x7E	MSB - LSB	Structură specifică frame-urilor individuale	Simplu byte

Delimitator start - Fiecare frame specific transmisiunii în modul API are un preambul specific pentru a ajuta dispozitivele la sincronizare. Acest preambul este reprezentat de o valoarea hexazecimală – 0x7E (126 în valoare zecimală). Ne folosim de acest lucru pentru a ştii când să citim şi la ce anume să ne aşteptăm în cadrul unei transmisii de date.

Lungime cadru – Următorii 2 bytes recepţionaţi după preambul ne indică lungimea totală a cadrului transmis, respectiv recepţionat – în majoritatea cazurilor, MSB (Most Significant Byte) are valoarea 0, în timp ce LSB (Least Significant Byte) conţine întreaga lungime. Atunci când se folosesc cadre cu o dimensiune mai mare, se poate modifica şi valoarea MSB. Totuşi, în cadrul ZigBee se recomandă să nu se folosească cadre de dimensiuni foarte mari, ci un număr mai mare de cadre.

Date (payload) – În această parte a cadrului sunt conţinute datele relevante în vederea prelucrării şi luării deciziei. Conţinutul acestui câmp reprezintă ceea ce anume ne dorim să transmitem de fapt între două noduri.

Checksum (verificator) – Acest ultim byte din cadrul unui frame reprezintă modalitatea prin care ne asigurăm că datele au fost transmise în întregime şi nu au existat pierderi pe parcurs. Valoarea scrisă în acest câmp este obţinută printr-o formulă de însumare a tuturor byte-urilor de dinainte – din acest rezultat se păstrează doar ultimii 8 biţi, iar rezultat se scade din hexazecimalul 0xFF.

Înainte de a transmite un cadru de date, orice nod XBee calculează această valoare şi o scrie în ultimul câmp disponibil. La destinaţie, se recalculează această valoare, iar dacă cele două coincid, atunci se poate concluziona că datele au fost transmise în întregime, fără a exista pierderi.

Respectând structura generală a cadrului prezentată mai sus, există mai multe tipuri de mesaje folosite pentru a trimite sau primi informaţii de la un nod radio XBee. Există un număr foarte mare de cadre specifice protocolului, dar în continuare ne vom concentra doar pe cele mai importante.

Byte-ul corespunzător tipului de cadru de date ne indică ce fel de cadru API am primit/trimitem şi ne permite să îl analizăm – de exemplu un cadru de date de tip 0x08 ne indică faptul că urmează o comandă de tip AT.

Tabel 3.4 – Structura generală a unui frame de comandă [13]

Tipul de frame	0x08	0x09	0x17	0x88	0x8A	0x10	0x8B	0x90
Descriere	Comandă AT	Comandă AT (pusă în coadă)	Cerere pentru comandă la distanță	Răspuns pentru o cerere de comandă AT	Status modem	Cerere TX	Răspuns TX	Răspuns RX

CAPITOLUL 4
PROIECTAREA UNUI SISTEM DE APRINDERE DE LA DISTANȚĂ A LUMINII

În cadrul acestui proiect ne propunem să controlăm intensitatea luminoasă dintr-o încăpere, la distanță, folosind protocolul ZigBee. Urmează să prezentăm implementarea în două moduri:

- manuală – cu ajutorul unui potențiometru se va regla la distanță intensitatea luminii
- automată – folosind un fotorezistor, vom măsura intensitatea luminoasă dintr-o anumită zonă și vom aprinde luminile atunci când este depășit un anumit prag.

Transmisia de date în cadrul acestor proiecte se va face de asemenea în două moduri diferite, pentru a putea ilustra posibilitățile oferite de către protocol și de către modulele XBee ce urmează să fie utilizate – în primă fază vom face transmisia de date ajutându-ne în totalitate de modulele Arduino Uno, ulterior trecând la transmisiune directă între noduri.

4.1 APRINDEREA MANUALĂ A SISTEMULUI DE ILUMINAT

În cadrul acestui scenariu vom avea nevoie de conectarea modulelor XBee la o placă de dezvoltare Arduino, pentru a se putea face transmisia datelor citite de la un potențiometru, respectiv aprinderea unui sistem de iluminat. În cadrul scenariului nostru ne vom limita la aprinderea unor LED-uri conectate pe plăcuțele de dezvoltare (breadboard), dar logica implementată poate să fie păstrată și pentru un sistem de dimensiuni mai mari.

În cadrul proiectării acestui sistem vom folosi următoarele componente:

- ✓ 2 module XBee Series 2 – unul dintre ele configurat în mod de coordonator AT, în timp ce al doilea în mod endpoint AT
- ✓ 2 plăcuțe intermediare pentru conectarea modulelor XBee la breadboard – spațierea dintre pinii modulelor XBee este de doar 2mm, în timp ce breadboard-urile standard au o spațiere între puncte de 2.54mm (1 inch).
- ✓ 2 breadboard-uri
- ✓ Un potențiometru
- ✓ Un modul de conectarea a plăcuțelor XBee la calculator în vederea configurării acestora
- ✓ 1 Rezistor
- ✓ 2 plăcuțe de dezvoltare Arduino Uno.

Pentru a putea trece mai departe, vom prezenta pe scurt placa de dezvoltare Arduino Uno, împreună cu IDE-ul folosit pentru programarea sa (Arduino IDE).

4.1.1 Placa de dezvoltare Arduino Uno

Arduino Uno este una dintre cele mai populare plăci de dezvoltare folosite în cadrul dezvoltării de proiecte electronice, întrucât este ușor de configurat, are un număr destul de mare de pini și este compatibilă cu un număr foarte mare de shield-uri, care permit adăugarea diverselor funcționalități.

Printre avantajele folosirii plăcuței Arduino Uno se numără faptul că nu este necesară scrierea manuală a prea multor funcții, întrucât, foarte multe dintre ele, deja există scrise în librăriile Arduino IDE și este necesar doar includerea lor în programele ce sunt dezvoltate. Acest lucru duce mai departe la ușurința utilizării unui număr foarte mare de shield-uri, fără a se pune probleme legate de rescriere a bibliotecilor sau de incompatibilități cu componente externe.

De asemenea, după ce am încărcat secvența de cod, aceasta rămâne salvată în cadrul memoriei flash chiar și după ce nu mai avem alimentare. În momentul în care realimentăm placa nu mai este nevoie să ne reconectăm și să retransmitem codul întrucât el este deja acolo și este rulat automat.

Ca și dezavantaje putem enumera frecvența destul de scăzută a microprocesorului și memoria nu foarte cuprinzătoare de tip flash și EEPROM. Un alt amănunt care nu se pliază foarte bine conceptului de Internet of Things este acela că placa nu vine cu un adaptor Ethernet inclus și trebuie achiziționat separat shield-ul potrivit.

- ✓ Specificații tehnice:
- ✓ Microcontroler ATmega328
- ✓ Memorie flash: 32KB
- ✓ Frecvență procesor: 16 MHz
- ✓ Memorie SRAM: 2KB
- ✓ Memorie EEPROM: 1KB
- ✓ Pini digitali I/O: 14(dintre care 6 pot oferi output PWM)
- ✓ Pini de intrare analogi: 6

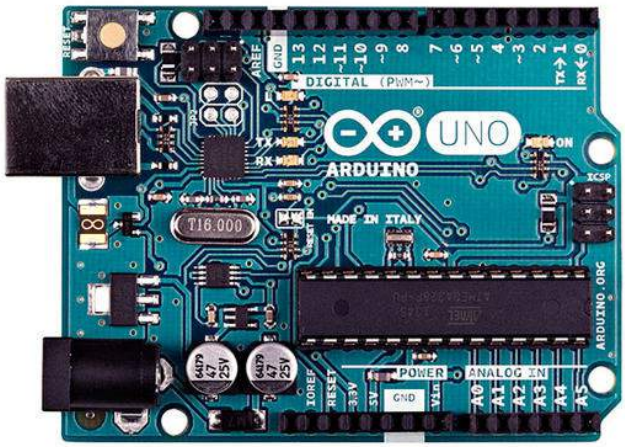


Figura 4.1 – Plăcuța de dezvoltare Arduino Uno

4.1.2 Programarea Arduino Uno

Așa cum am precizat anterior, parte a conceptului Arduino este aceea care susține că orice individ, având cunoștințe minime sau în dezvoltare legate de programare și electronică să poată să folosească tehnologia oferită de către plăcuțele Uno, Mega, Leonardo etc. în scopuri personale sau pentru a-și ușura stilul de viață.

Pentru ca acest lucru să fie posibil este nevoie de un sistem de operare al plăcilor care să poată să înțeleagă limbaje de programare simple și care să nu aibă nevoie de includerea a prea multor biblioteci, cel puțin la început. De asemenea, modul de comunicare dintre utilizator și hardware trebuie să fie destul de simplu.

Software-ul care rulează pe plăci îndeplinește toate aceste condiții, iar cu ajutorul Arduino IDE poate folosi orice limbaj de programare(C/C++/Java etc.) pentru a scrie programele. Compilatorul inclus în cadrul aplicației poate înțelege toate aceste limbaje, pe care le traduce în limbaj mașină și le trimite către placa de dezvoltare. Un plus al Arduino IDE mai este faptul că, spre diferență de alte programe de verificat și compilat cod, funcționează fără nicio problemă pe orice sistem de operare care rulează pe calculatoare: Windows, Mac OS X sau diverse versiuni de Linux.

Arduino IDE vine, de asemenea, integrat cu o librărie software "Wiring", care conține foarte multe proceduri și funcții des utilizate pentru citirea informațiilor de la pin-urile plăcii sau pentru trimiterea de semnale către acestea.

Cele mai simple programe în limbajul C/C++ ce pot fi rulate pe Arduino, conțin două funcții de bază, obligatorii, ce sunt compilate și legate între ele în cadrul unei alte funcții main():

- setup() – o funcție ce rulează o singură dată la pornirea programului și inițializează parametrii de lucru ai programului ce urmează să fie pus în aplicare
- loop() – o funcție ce este reluată și rulată până când placa de dezvoltare nu mai este alimentată

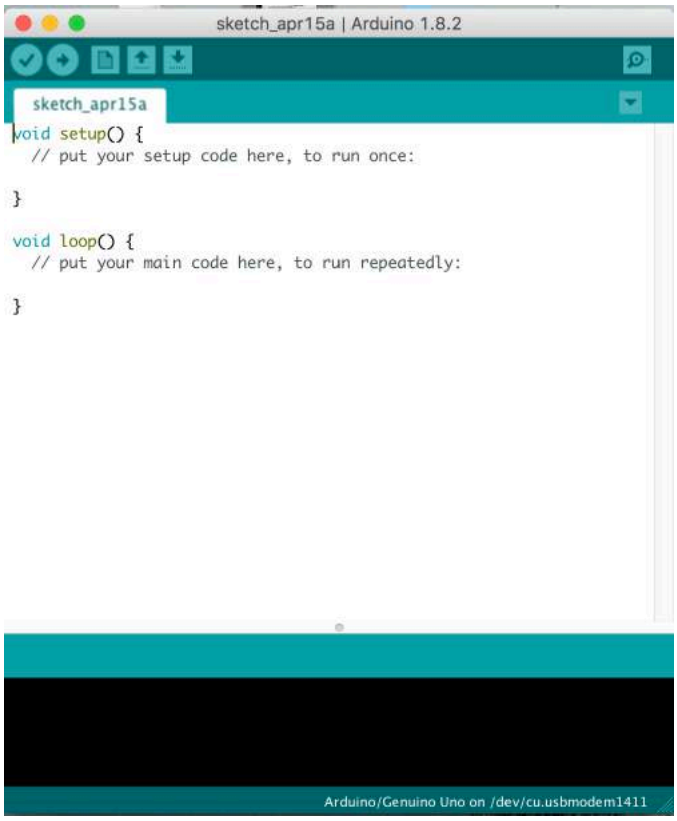


Figura 4.2 – IDE-ul pentru programarea Arduino

4.2.1 Schema electrică a proiectului

Conectarea modulelor XBee nu se va face direct pe breadboard, din cauza spațierii pinilor. Vom folosi un breakout-board pentru conectarea intermediară dintre cele două. Acest lucru nu a putut fi explicat în schema electrică realizată cu ajutorul programul fritzing.

Pentru partea de receptor (sistemul efectiv de iluminare) avem următoarea schemă electrică:

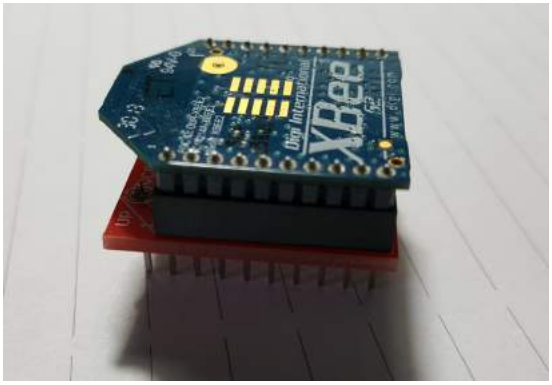


Figura 4.3 – Modulul XBee conectat pe break-out board

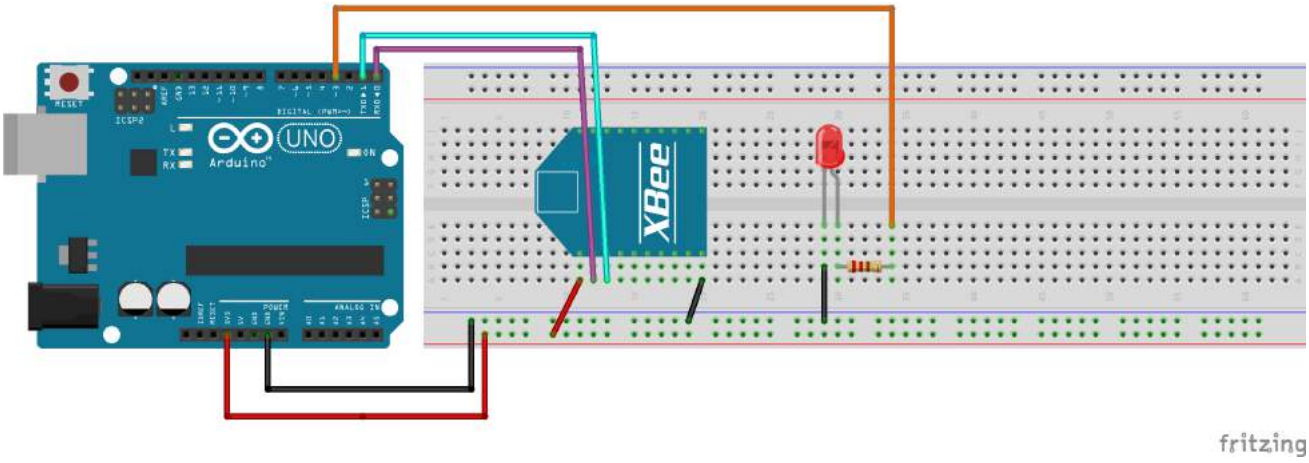


Figura 4.4 – Schema electrică receptor luminos

După cum se poate observa pe schemă, alimentăm nodul XBee la tensiune de 3.3V (acesta nu va funcționa la o tensiune mai ridicată – cum ar fi cea de 5V) – prin conectarea alimentării la pinul 1. Legăm și pinul de masă (ground) al modului XBee – pinul 10 în mod corespunzător pe breadboard.

LED-ul pe care dorim să îl aprindem de la distanță îl vom conecta la un pin PWM (Pulse Width Modulation) – pinul 3 în cazul nostru, pentru a putea transmite tensiuni variabile, care să ne reprezinte aprinderea la diverse intensități.

Arduino Uno trebuie să aibă pinii de transmisie și recepție a datelor (TX, respectiv RX – 1 și 0) conectați la pinii echivalenți ai modulului XBee (pinii 2 și 3).

Folosim de asemenea și un rezistor pentru a nu conecta în mod direct LED-ul la pinul de alimentare, evitând în felul acesta arderea diodei.

În continuare prezentăm schema electrică a emițătorului (controlerul sistemului de iluminat).

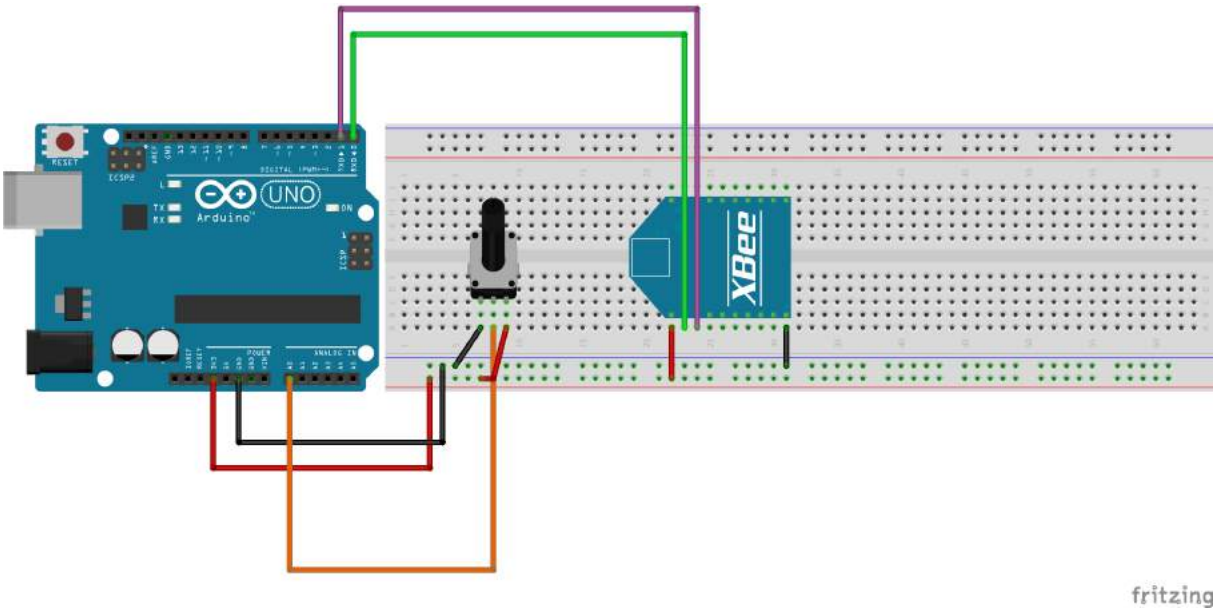


Figura 4.5 – Schema electrică emițător cu potențiometru

După cum se poate observa, doi dintre pinii potențiometrului se conectează la alimentare, respectiv masă, în timp ce cel de-al treilea este conectat la unul dintre pinii analogici ai plăcuței de dezvoltare Arduino Uno. Folosim un astfel de pin pentru a putea citi în mod corespunzător valorile trimise de către potențiometru.

Ca și în schema de mai sus, modulul XBee se conectează la Arduino Uno prin cei doi pini RX/TX, fiind alimentat la o tensiune de 3.3V.

4.2.2 Configurarea modulelor XBee

După cum am prezentat și în partea introductivă a lucrării, într-o rețea ce funcționează cu ajutorul protocolului ZigBee, avem nevoie de un coordonator unic. Acesta este cel care impune folosirea unui anumit canal de transmisiune, stabilește adresele în cadrul rețelei și primește toate datele de la end device-uri și de la routere. În cadrul rețelei noastre, topologia aleasă va fi ”perece” (pair), urmând să configurăm unul dintre moduri în modul Coordonator AT, iar cel de-al doilea în modul Endpoint AT.

În procesul de configurare a modulelor trebuie să ne asigurăm că setăm în mod corespunzător adresele destinație ale cadrelor (putem specifica adresele finale în acest scenariu întrucât avem o topologie de tip ”pereche”).

Pentru conexiunea între calculator și modulele XBee am folosit Tera Term – un client ce citește și afișează informațiile de pe un port serial într-o interfață ușor de citit.

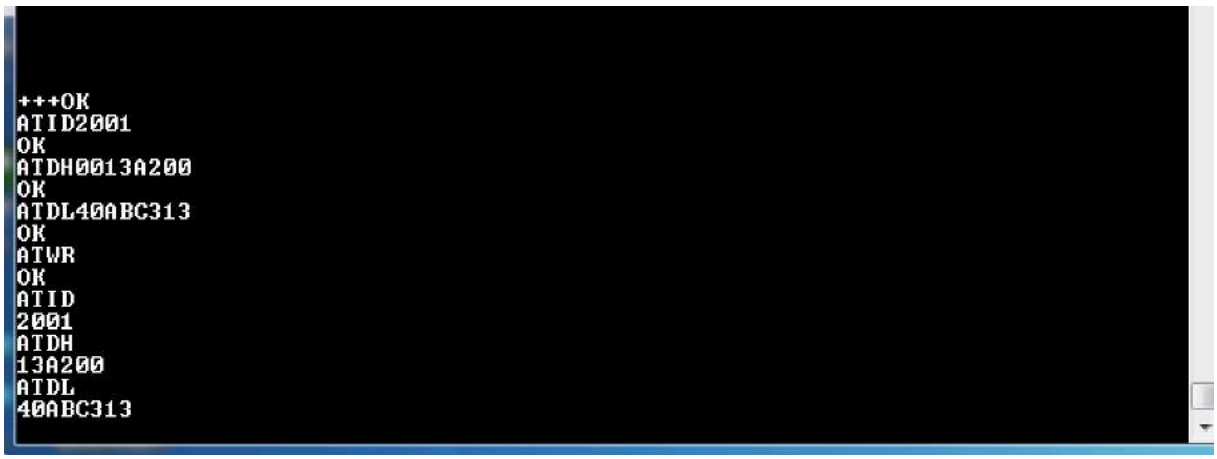


Figura 4.6 – Configurarea XBee a coordonatorului - controlerul intensității luminoase

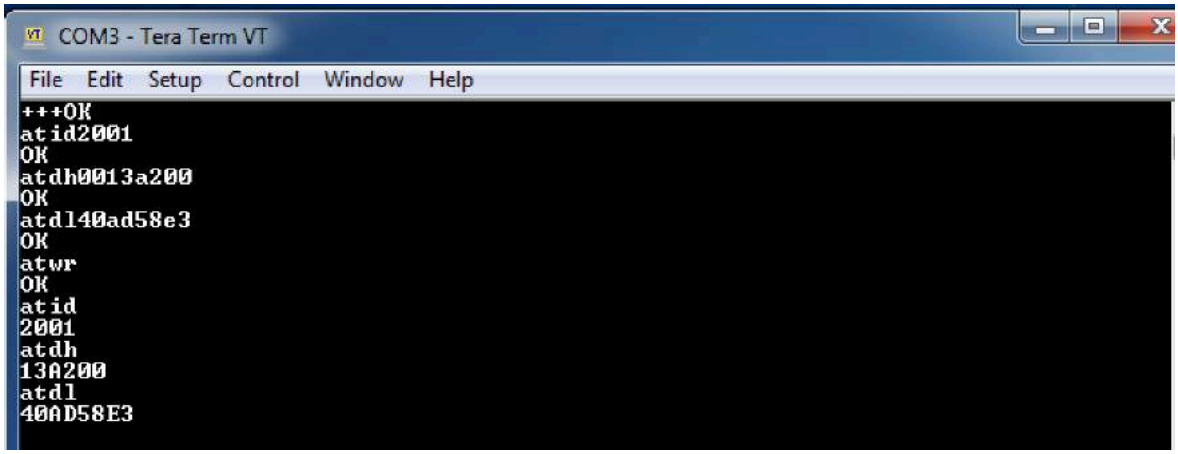


Figura 4.7 – Configurarea XBee a receptorului luminos

4.2.3 Configurarea modulelor Arduino

Secvențele de cod ce trebuie încărcate pe plăcuțele Arduino Uno pentru a se îndeplini funcționalitatea propusă în cadrul scenariului sunt prezentate în cadrul anexei 2 pentru coordonator, respectiv anexei 3 pentru receptor. În continuare voi explica codul prezentat, logica din spatele implementării și funcționalitatea efectivă.

Chiar dacă nu ne vom folosi încă de încapsularea datelor disponibilă în modul API de funcționare al nodurilor XBee, în cadrul routerului vom folosi un anumit tip de formatare a datelor, pentru a fi mai sigură transmisia efectivă de date . Mai exact, după cum se poate vedea și în anexa 2 valoarea citită de la pinul analogic, respectiv de la potențiometru, va fi înconjurată de paranteze unghiulare, respectând forma: <***valoare numerică***>. Ulterior, caracterele sub această formă sunt trimise rând pe rând în cadrul comunicației wireless.

La capătul celălalt al sistemului, primul caracter ”așteptat” pentru citire este reprezentat chiar de prima paranteză unghiulară – ulterior se începe stocarea valorii numerice primite, până la întâlnirea celei de-a doua paranteze unghiulare.

4.2 APRINDEREA AUTOMATĂ A SISTEMULUI DE ILUMINAT

În cadrul acestui proiect ne propunem să automatizăm procesul de aprindere a sistemului de iluminare, cu ajutorul unui senzor, reprezentat de către un fotorezistor. Mai exact, unul dintre modulele XBee va achiziționa în mod direct (fără o placă de dezvoltare Arduino) informații legate de luminozitatea dintr-un anumit punct – relevant ar fi de lângă un geam, pentru a fi monitorizată luminozitatea de afară, în timp ce un alt modul XBee se va ocupa de aprinderea sistemului, prin intermediul unei plăcuțe Arduino.

În cadrul proiectului vom folosi aproximativ aceleași componente ca în cadrului celui de mai sus, cu excepția faptului că vom elimina necesitatea uneia dintre plăcuțele Arduino și a potențiometrului, dar vom folosi în schimb un fotorezistor pentru măsurarea luminozității.

4.2.1 XBee Direct

Orice modul radio XBee este dotat cu capabilitatea de a colecta informații de la un număr de senzori și a transmite aceste informații în cadrul unei rețele, fără a se utiliza un microcontroler suplimentar. Pe lângă această funcționalitate, XBee mai oferă simple funcționalități de generare a semnalelor, pentru activarea diverșilor acuatori – de exemplu se pot transmite informații prin intermediul unui astfel de modul radio pentru a putea controla un motor, un sistem de iluminat sau alte lucruri asemănătoare – această funcție se regăsește în literatura de specialitate cu titlul de XBee Direct.

Există un număr de avantaje ce vin la pachet cu folosirea acestui concept – cele ce merita specificate sunt reprezentate de reducerea dimensiunii proiectului și a greutății (lucruri care sunt relevante pentru proiectele specifice Internet of Things). Pe lângă acestea, cel mai important avantaj este consumul de energie care este redus în mod semnificativ, prin eliminarea microcontrolerului. Alimentarea necesară unui modul radio XBee este de doar 3.3V, iar amperajul are valori semnificativ mai mici decât o placă Arduino, de exemplu.

Totuși, există și dezavantaje prezentate, cum ar numărul limitat de pini – pentru un microcontroler putem cu ușurință să adăugăm pini suplimentari sau să înlocuim microcontrolerul cu totul, în timp ce XBee se prezintă cu această limitare. Modulele din Series 2 nu au pini de ieșire analogici, ceea ce înseamnă că nu vom putea în niciun moment să aprindem un LED la valori intermediare, în mod asemănător proiectului de mai sus. Pe scurt, nu se pot lua decizii independente cu ajutorul acestor plăcuțe, ci doar să se transmită datele achiziționate sau cel mult să se schimbe starea unui pin.

XBee Series 2 vin încorporate cu un număr de 10 pini ce oferă o flexibilitate relativă proiectelor în care sunt folosite. Aceştia pot fi configuraţi atât ca şi intrări digitale pentru a putea urmări activitatea unui buton sau a unui întrerupător, dar şi ca ieşiri digitale pentru aprinderea unui LED sau controlului direct al unui motor.

În scenariile în care este necesar să se folosească surse de curent alternativ, se pot adăuga pinilor cu ieşiri digitale relee care să controleze instalaţii ce necesită alimentare mai mare. Primii patru pini din cei pe care îi avem la dispoziţie pot fi configuraţi pentru a citi date de tip analogic – cum ar fi datele generate de către senzorii de luminozitate, temperatură, forţa, accelerare, umiditate, gaz etc. Nu avem la dispoziţie pini de ieşire analogică, pentru a putea configura o ieşire de tip PWM în vederea controlării directe a unui motor sau a unui altfel de sistem asemănător, dar un firmware corespunzător acestei funcţionalităţi va fi lansat în curând.

4.2.2 Schema electrică a proiectului

Senzorul de lumină va fi conectat la primul pinul analogic al modulului XBee, ce va fi configurat în mod analogic de intrare. Conectăm fotorezistorul în mod indirect la alimentare, folosind o rezistenţă de valoare dublă a acestuia (în cazul nostru, 20kΩ - vom folosi de fapt două rezistenţe înseriate). Folosim această măsură pentru a modera valoarea generată de către senzor, în vederea prelucrării la destinaţie.

Alimentarea se va face, pentru senzor, cu ajutorul unor baterii. În cadrul proiectului mi-am propus înlocuirea acestor baterii cu panouri fotovoltaice care să furnizeze energia necesară alimentării modulului XBee.

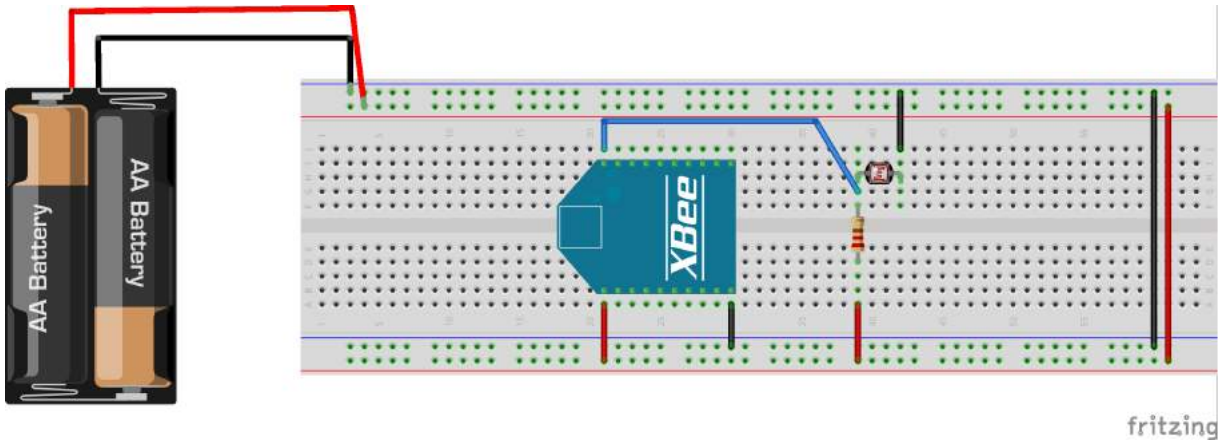


Figura 4.8 – Schema electrică a senzorului de luminozitate

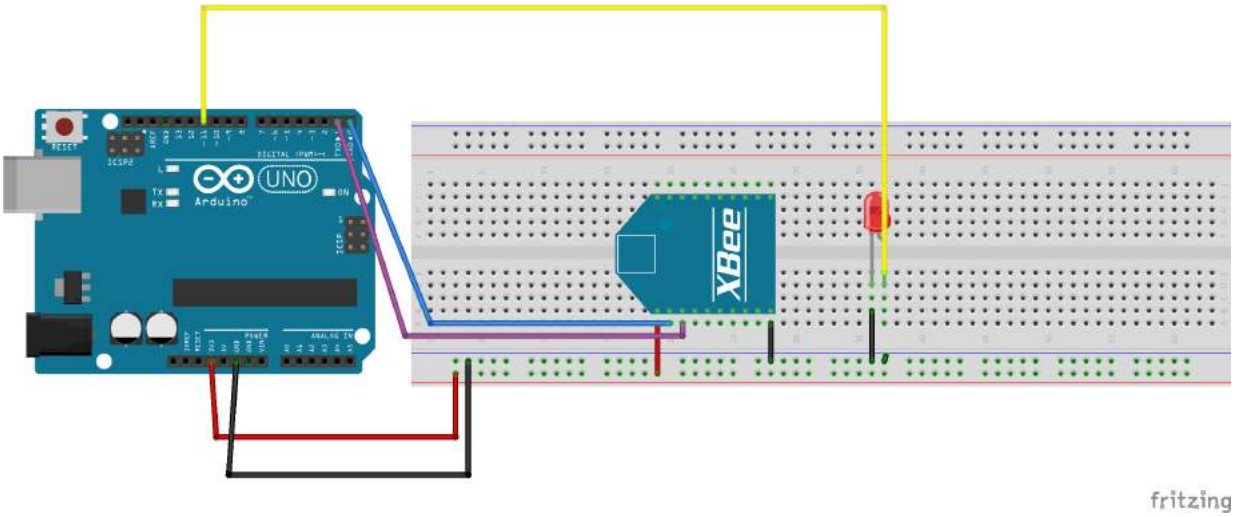


Figura 4.9 – Schema electrică a LED-ului aprins

Sistemul de iluminat va fi în continuare conectat la o placă de dezvoltare Arduino, datorită motivelor prezentate în subcapitolul anterior. În acest caz, alimentarea microcontrolerului se poate face într-un mod diferit (nu direct de la calculator printr-un cablu serial), pentru a oferi mobilitate sistemului de iluminare, iar în locul unui LED pot fi conectate benzi de LED-uri ce pot fi alimentate cu ajutorul unui releu.

4.2.3 Configurarea modulelor XBee

- Configurarea senzorului de luminozitate

În cadrul acestui scenariu, modulul XBee conectat la senzorul de luminozitate va fi configurat ca și router, fiind responsabil cu trimiterea mesajelor către coordonator. Îi vom configura cele două părți ale adresei destinație, DH/DL cu cele două părți corespunzătoare ale nodului coordonator. În felul acesta, toate mesajele vor fi direcționate către acesta.

Vom configura de asemenea pinul 0 în mod analogic de achiziție a datelor și vom seta frecvența de achiziție la 100ms (în hexazecimal – 0x64) – echivalent cu 10Hz.

```
+++OK
atid2001
OK
atdh0013a200
OK
atdl40ad58e3
OK
atjvr1
OK
atd02
OK
atir64
OK
atwr
atwr
OK
atid
2001
atdl
40AD58E3
atdl
40AD58E3
atjvr
OK
atjvr1
OK
```

Figura 4.10 – Configurarea XBee senzor de luminozitate

Comanda ATIR64 este cea cu care am setat frecvența de eșantionare, ATD02 este cea cu care am setat pinul 0 în mod analogic, iar cu ajutorul comenzii ATJVR1 m-am asigurat că nodul va încerca reconectarea la coordonator în cazul în care, pentru o perioada de timp, se pierde legătura dintre cei doi.

- Configurarea sistemului de iluminat

În acest scenariu, baza – respectiv sistemul de iluminat de sine stătător, va fi configurat în mod API. Acest lucru înseamnă că nu vom mai putea seta informațiile asemănătoare celor de mai sus cu ajutorul unui terminal prin care să trimitem comenzi seriale, ci vom folosi direct programul XCTU pentru a transmite toate informațiile relevante.

În acest caz, configurările trebui să includă selectarea aceluiași PAN ID ca și routerul (2001 în cazul nostru) și configurarea adreselor destinație. Ultimul lucru nu este obligatoriu, întrucât în scenariul curent, coordonatorul nu trebuie decât să primească mesaje, nu să și trimită.

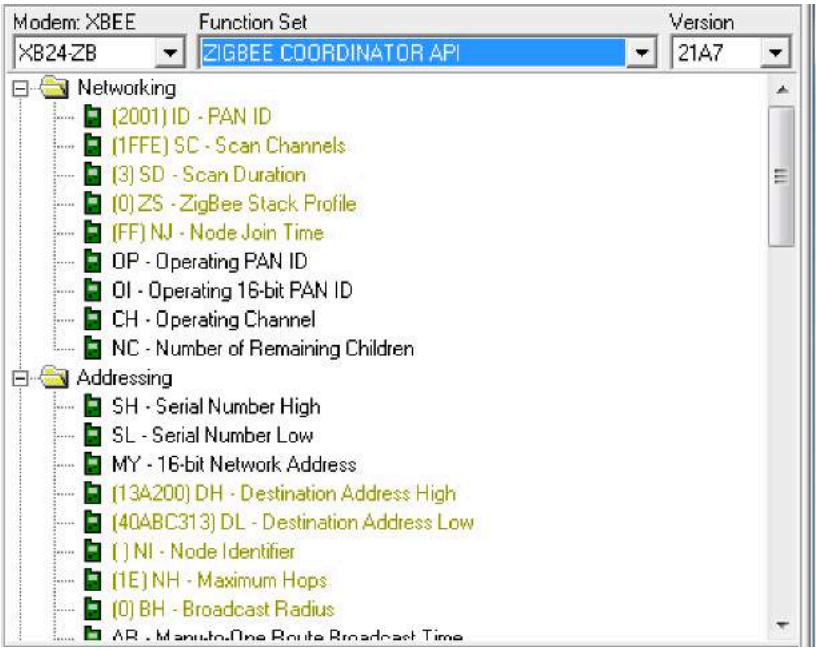


Figura 4.11 – Configurare XBee sistem de iluminat

4.2.4 Configurarea modului Arduino

În cadrul acestui proiect, datorită funcționalii XBee Direct, am putut folosi o singură plăcuță de dezvoltare Arduino, cea responsabilă cu aprinderea sistemului de iluminare, reducând în felul acesta și consumul total de energie dar și costul de implementare total.

După cum se poate observa în anexa 4, în programului încărcat pe plăcuța Arduino Uno, începem prin inițializarea pinilor și stabilirea rolurilor acestora – am folosit pe lângă LED-ul ce se va aprinde în funcție de valoarea primită legată de intensitatea luminoasă din jur, un alt LED cu scopul de a face troubleshoot legat de primirea de date – acesta se va aprinde cu o frecvență de o secundă doar în momentul în care se citesc date pe legătura serială.

Datele preluate de modulul XBee nu sunt prelucrate sub nicio formă, fiind doar transmise către pinii microcontrolerului, acolo unde se caută valoarea hexazecimală **0x7E,** ce reprezintă începutul unei transmisii de date. Ulterior, în cadrul aceluiași pachet se așteaptă a fi conținută valoarea ce reprezintă intensitatea luminoasă din proximitatea fotorezistorului – cu cât aceasta va fi mai mică, cu atât vom considera că este necesar să aprindem LED-ul la o intensitate mai puternică. Am stabilit 2 praguri și 3 nivele de aprindere a acestuia, dar în funcție de necesități și de sursa de luminozitate aleasă, putem realiza o fragmentare mai mare a acestor intervale.

CAPITOLUL 5
PROIECTAREA UNEI SONERII WIRELESS PENTRU UN BLOC DE APARTAMENTE

Îmi propun în continuare să folosim modulele XBee împreună cu plăcuțele de dezvoltare Arduino pentru a ilustra un scenariu în care scopul principal este reducerea numărului de cabluri folosite. Dacă în cazul sistemului de iluminare îmi propusesem și o mobilitate ridicată, în acest caz ne dorim să eliminăm numărul de fire trase prin pereții unei clădirii, păstrând în continuare funcționalitatea soneriei clasice.

Mă voi folosi tot de protocolul ZigBee pentru a configura la intrarea în clădire un nod XBee care să joace rolul de router, achiziționând date cu ajutorul unei plăcuțe Arduino, de la butoanele apăsate. În cadrul clădirii vom configura un coordonator care să primească datele trimise de la intrare și împreună cu o altă placa Arduino Uno, să sune soneria corespunzătoare apartamentului pentru care a fost apăsat butonul.

Acest proiect a fost testat pentru 3 apartamente în cadrul unui imobil mic de înălțime, dar cu foarte mare ușurință poate să fie reconfigurat și scalat și la un număr mai mare de apartamente, respectiv la un bloc de dimensiuni mai mari.

5.1 SCHEMA ELECTRICĂ A PROIECTULUI

După cum se poate observa în cadrului figurii 5.1, am atașat plăcuței Arduino două butoane, câte unul pentru fiecare apartament în parte pentru care am conceput sistemul de sonerie. În funcție de butonul care urmează să fie apăsat, se va genera un caracter diferit, ce va fi trimis către modulul XBee cu ajutorul pinilor seriali de transmisie, respectiv citire a datelor.

În cadrul modului XBee, acest caracter va fi încapsulat într-un pachet diferit, corespunzătorul modului API de funcționare, urmând să fie ulterior trimis în rețea către coordonatorul ce așteaptă mesajele și cunoaște încapsularea folosită.

Pentru a nu cauza probleme asupra plăcuței Arduino, am conectat indirect butoanele la pinii acesteia, folosind o rezistență de 1kΩ. În felul acesta, tensiunea ajunge în continuare la pinul pe care se face citirea datelor, dar ajunge cu o valoarea mai mică – fără folosirea acestei rezistențe, am fi riscat să apară probleme la utilizarea pe termen lung a sistemului.

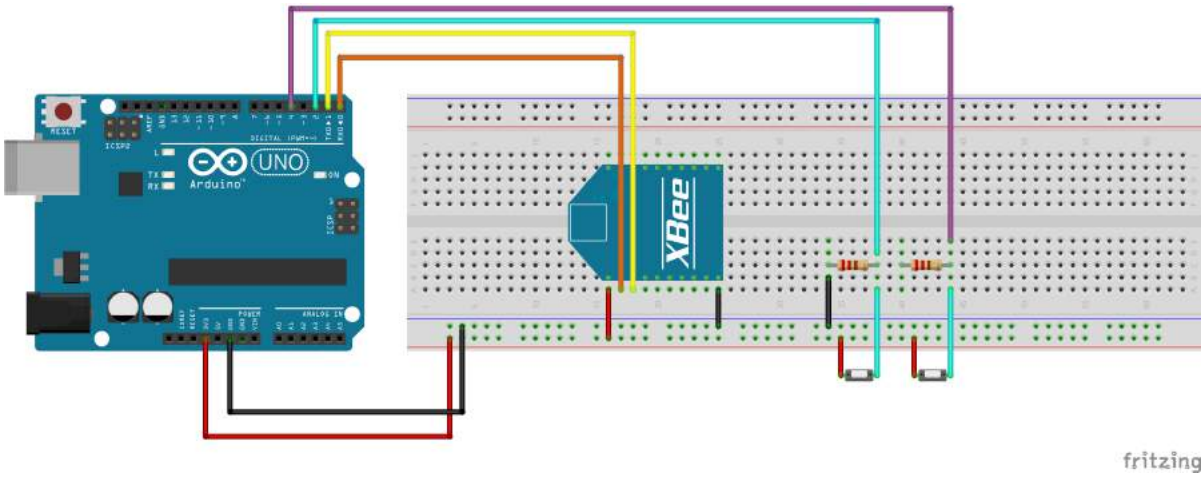


Figura 5.1 – Routerul de la intrarea în clădire

În continuare, prezentăm schema electrică a coordonatorul, cel care are rolul de a suna soneria de la apartamentul corespunzător butonului apăsat la intrare. Aceasta se poate observa și în cadrul figurii 5.2.

Semnalul primit de la router este captat cu ajutorul modului XBee, ce recunoaște încapsularea specifică modului API de funcționare și extrage doar informațiile de interes - și anume caracterul corespunzător butonului apăsat. Acesta este transmis mai departe către plăcuța Arduino Uno cu ajutorul pinilor Tx/Rx (pinii 0 și 1 de pe plăcuță, respectiv 1 și 2 de pe XBee), unde se face citirea mesajului și sunată soneria corespunzătoare.

Pentru a demonstra funcționalitatea nu am folosit mai multe sonerii, ci, după cum se poate observa și în cadrul anexei 6, respectiv a codului de pe plăcuța Arduino a coordonatorului, am folosit o singură sonerie, pentru care se vor emite sunete diferite, în funcție de butonul apăsat.

De menționat faptul că numărul de pini analogici de pe plăcuța Arduino Uno ne permite să extindem cu ușurință numărul de sonerii folosite, respectiv a apartamentelor care să se folosească de acest sistem. În cazul în care dorim să extindem și mai mult proiectul, ne putem folosi de alte plăcuțe din familia Arduino, care au și mai mulți pini disponibili – Leonardo, Mega etc.

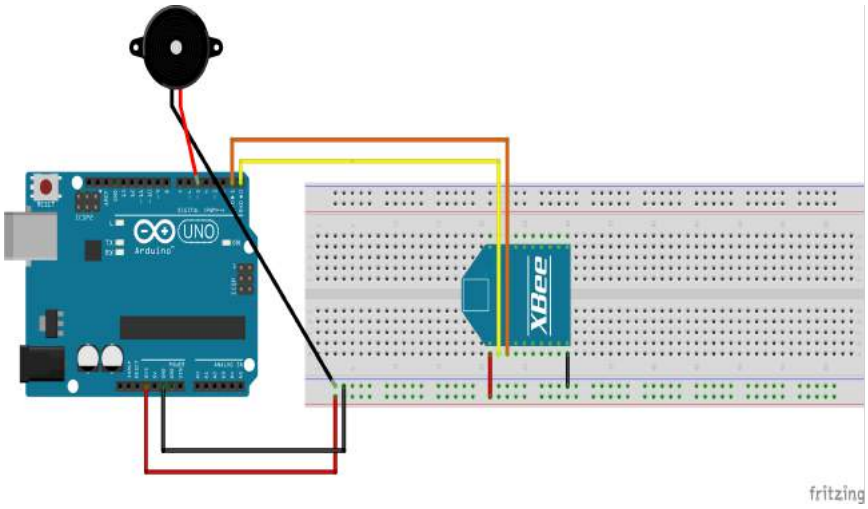


Figura 5.2 – Coordonatorul responsabil cu soneria

5.2 Configurarea modulelor XBee

Din punct de vedere al configurării modulelor XBee, setările sunt asemănătoare în ambele părți ale proiectului, atât la router cât și la coordonator. Mai mult, acestea se asemanănă cu cele prezentate în partea de teorie, legată de configurarea adreselor destinație – fiind vorba de o topologie simplă, de tip pereche, putem configura adresele destinație clare pentru ambele noduri, eliminând în felul acesta pericolul interceptării datelor de către o terță parte.

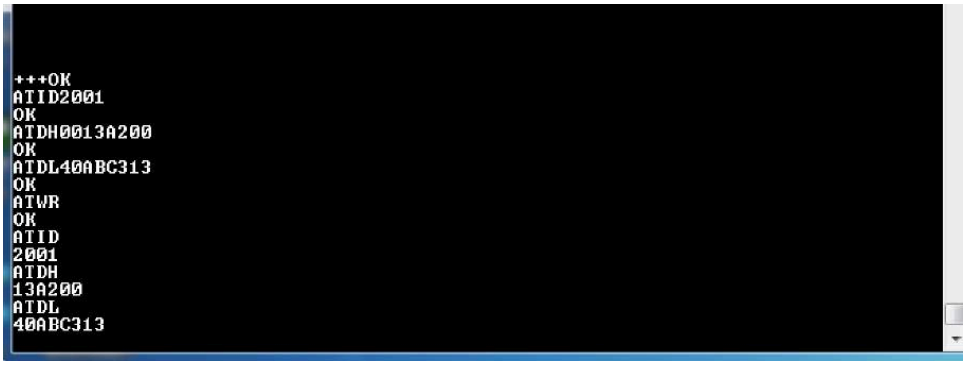


Figura 5.3 – Configurările XBee ale sistemului de sonerie

5.3 Configurarea modulelor Arduino

După cum este prezentat și în cadrul anexei 5, se începe prin inițializarea pinilor ce vor fi folosiți pentru intrare – butoanele, respectiv pentru ieșire – LED-urile care oferă un feedback în urma apăsării butoanelor. De asemenea, inițializăm caracterele corespunzătoare fiecărui buton în parte, care vor corespunde apartamentelor ce se vor conecta la modulul de coordonator.

În momentul în care este citită tensiune de pe unul dintre butoane, este transmis pe legătura serială corespunzătoare pinilor Tx/Rx, către XBee caracterul corespunzător acelui buton. Acest caracter este încapsulat într-un pachet de date wireless ce este trimis în mod direct către punctul de acumulare al datelor, reprezentat de coordonator.

La capătul celălalt, sunt așteptate, cu o întârziere de o secundă caracterele ce vor indica faptul că trebuie sunată o anumită sonerie. Am introdus un buffer pentru cazul în care butoanele sunt apăsate în mod continuu sau pentru cazul în care se apasă foarte rapid pe mai multe butoane. În felul acesta am reușit să evit aglomerarea portului serial și respectiv blocarea acestuia în anumite situații.

CAPITOLUL 6
PROIECTAREA UNUI SISTEM WIRELESS DE MONITORIZARE A TEMPERATURII

În cadrul ultimului proiect practic prezentat în cadrul lucrării curente mi-am propus să explorez capacitate de transmitere a datelor în timp real cu ajutorul protocolului ZigBee, respectiv achiziția acestor date de la mai multe noduri, în vederea afișării acestora către un utilizator. Mă voi folosi în continuare de plăcuțele Arduino de dezvoltare și de modul API de funcționare a nodurilor XBee.

Pentru implementarea proiectului am folosit pe lângă modulele XBee și plăcuțele de dezvoltare specificate anterior, senzorul de temperatură cu circuite integrat, Maxim Integrated DS18B20. Acesta prezintă un avantaj asupra altor senzori de temperatură din gama sa, având nevoie de un singur pin pentru a transmite informațiile în vederea stocării și analizării acestora. Precizia acestuia de ±0.5 °C este mai mult decât suficientă pentru mediul casnic de lucru, ce reprezintă ținta dezvoltării curente. De asemenea, am utilizat un LCD clasic 16 x 2 pentru prezentarea informațiilor legate de temperatură. Pentru aceste două componente există deja biblioteci create, ce sunt ușor de important și utilizat în cadrul dezvoltării cu IDE-ul Arduino. Mai mult decât atât, structura curentă a proiectului ne permite o dezvoltare ulterioară care să includă senzori pentru mai mulți parametrii – umiditate, nivel de fum, nivel de gaz etc.

Ca și dezvoltare ulterioară a acestui proiect mi-am propus includerea unei acțiuni separate a microcontrolerului la anumite temperaturi – pornirea aerului condiționat și configurarea acestuia la o temperatură dorită. Această acțiune urmează de asemenea să fie implementată în mod wireless, folosindu-mă de protocolul ZigBee la capacitatea sa maximă.

6.1 SCHEMA ELECTRICĂ A PROIECTULUI

După cum se poate observa și în figurile 6.1 și 6.2, unul dintre senzorii de temperatură a fost conectat în mod direct la bază, în timp ce un altul este conectat doar wireless la aceasta putând fi plasat într-o rază de acoperire de aproximativ 35-40 de metri. Am ales să folosesc această configurație pentru a simplifica demonstrația practică, dar în realitate, ambii senzori puteau fi conectați doar wireless la bază, unde să se afișeze temperaturile măsurate.

Pentru configurarea bazei, unde se achiziționează și afișează datele, am profitat de faptul că breadboard-ul este format din 2 părți ce sunt deconectate la mijloc, permițându-mi astfel alimentarea atât a modului XBee cât și a ecranului LCD pe care am realizat afișarea. Am avut nevoie de acest lucru întrucât modului XBee trebuie alimentat la 3.3V, în timp ce pentru o bună funcționare a ecranului LCD este recomandată o tensiune constantă de 5V.

Cu ajutorul unui potențiometru la LCD se reglează intensitatea scrisului ce apare pe LCD, în timp ce cu ajutorul unei rezistențe de 220Ω ne asigurăm că tensiune ce alimentează lumina de fundal (backlight) nu este prea mare pentru a arde circuitul ecranului.

Senzorul DS18B20 este conectat la o alimentare de 3.3V, iar pentru pinul său de date (cel din centru) folosim un divizor de tensiune pentru a suprasolicita placa Arduino Uno, conducând la probleme tehnice asupra acesteia. Acesta trimite datele către plăcuță, care mai departe, cu ajutorul pinilor Tx/Rx le trimite către nodul XBee.

Datele sunt ulterior plasate în rețea și trimise către coordonator.

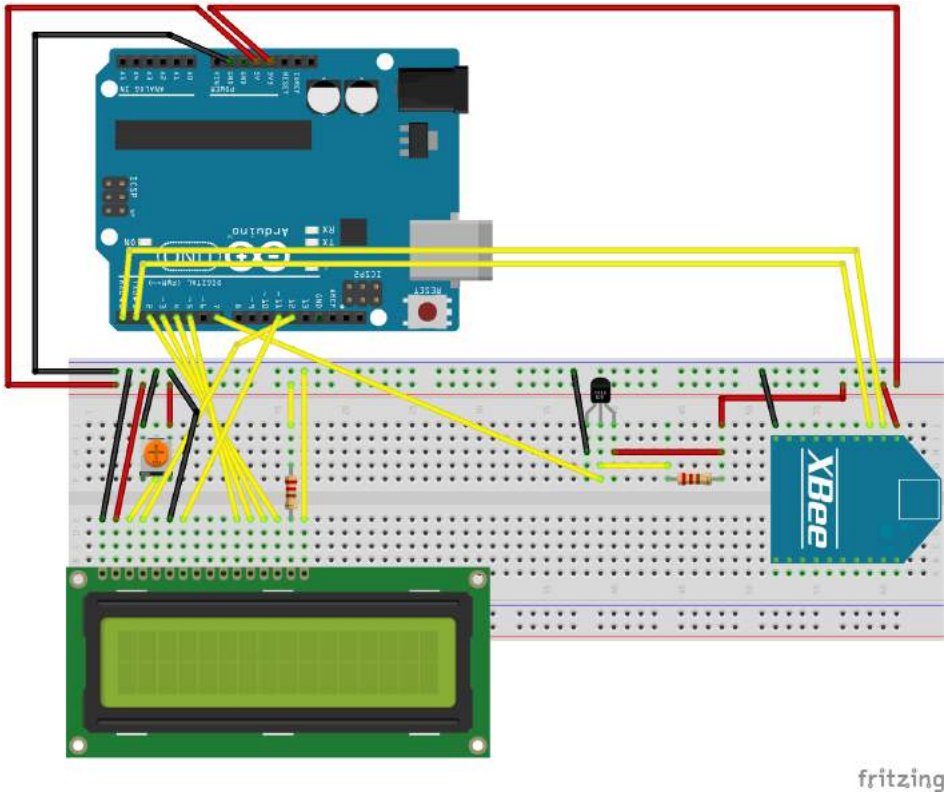


Figura 6.1 – Baza sistemului și primul senzor

Configurația senzorilor ce sunt plasați la distanță (remote) este asemănătoare cu cea din partea de bază, cu excepția faptului că nu vom mai avea si ecranul LCD conectat la Arduino. Folosirea

microcontrolerului în această situație este mai degrabă o restricție dată de modul în care senzorul de temperatură ales prelucrează datele – mai exact, este nevoie directă de librăria acestuia pentru a fi interpretate datele. Dacă am fi folosit un alt senzor, atunci am fi putut transmite datele fără acest intermediar, într-un mod asemănător celui abordat la primul proiect (unde datele de la fotorezistor erau captate și trimise direct de către nodul XBee). În acest caz, prelucrarea datelor s-ar fi făcut în totalitate la bază.

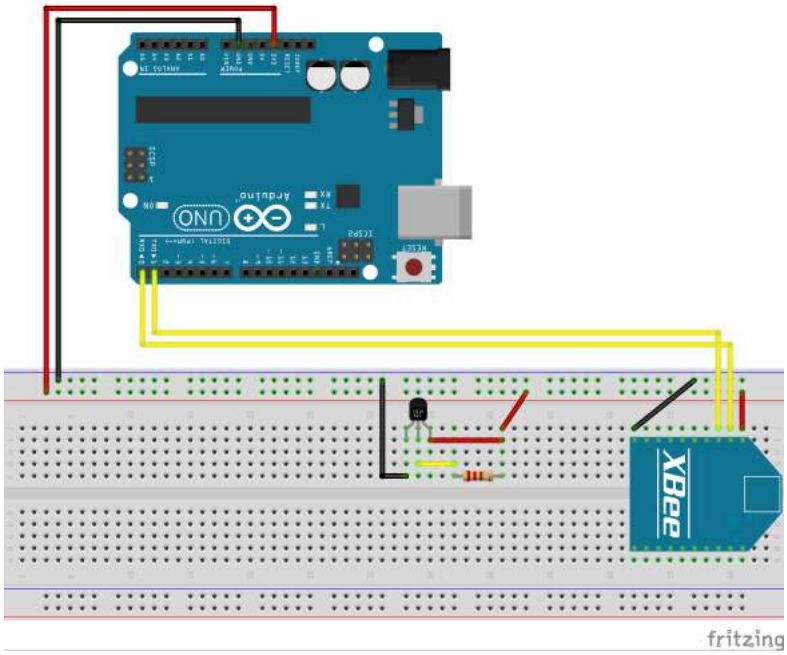


Figura 6.2 – Senzor de la distanță al sistemului

6.2 Configurarea modulelor XBee

Din punct de vedere al configurărilor modulelor wireless, nu am avut lucruri noi introduse, setările făcute anterior în cadrul proiectelor fiind asemănătoare cu cele introduse aici. Mai exact, avem nevoie de configurare din punct de vedere al adreselor destinație al modulelor ce conțin senzorii de la distanță – baza nu are nicio responsabilitatea de a trimite date către nimeni din cadrul rețelei, acest punct reprezentând un data sink – practic avem doar acumulare de date ce trebuie prelucrate și afișate utilizatorului.

În cazul în care am fi folosit senzori care să îndeplinească alte roluri sau un alt senzor de temperatură, care nu ar fi avut nevoie de o librărie specifică pentru prelucrarea datelor, am fi putut conecta pinul de date al senzorului de temperatură direct la unul dintre pinii analogici ai modului XBee.

6.3 Configurarea modulelor Arduino

După cum am specificat încă din introducere și după cum se poate observa în cadrul anexelor 7 și 8, vom avea nevoie de bibliotecile create pentru o mai simplă interacționare cu ecranul LCD și cu senzorul de temperatură ales – acestea vor fi importante încă de la început. De asemenea, este nevoie de inițializare a celor două componente, încă de la începutul programului, pentru a specifica ce pini vor fi folosiți pentru funcționalitățile aduse – pentru ecranul LCD vom folosi pinii 2, 3, 4, 5, 11, 12, în timp ce pentru senzorul DS18B20, în cazul ambilor senzori, selectat pinul 7 pentru achiziția de date.

În partea de setup() a ambelor părți a proiectelor, inițializăm legătura serială cu calculatorul, respectiv cu modulele XBee, setând o viteză de transmisie a datelor (baud rate) de 9600 bps si pornim achiziția de date de la senzorul de temperatură, respectiv inițializăm LCD-ul.

Atât în cazul senzorului plasat la distanță, cât și în cazul bazei, unde se achiziționează datele, prefer stocarea, transmisia și afișarea datelor sub formă de Stringuri – am ales această abordare pentru a fi mai ușor pe viitor de stocat datele de la alți senzori. De asemenea, a reprezentat cea mai bună soluție pentru problema întâmpinată în momentul în care trimiteam date numerice pe legăturile seriale ale Arduino, acesta realizând o conversie implicită în cod ASCII a acestora.

În cadrul ambilor senzori, datele sunt achiziționate în partea de loop() a programului prin apelul funcției **getTempCByIndex()** – care returnează valoarea achiziționată la un anumit moment de timp. La această valoare concatenăm simbolul de grade Celsius, după care rezultatul îl transmitem către modulul XBee, respectiv le afișăm. De menționat că în cadrul codului prezentat, afișarea se face și pe monitorul serial, pentru o mai bună înțelegere a rezultatelor obținute în ambele puncte.

Pentru partea de afișare a datelor, procesul este următorul – setăm prima linie a ecranului pentru afișare, realizăm achiziția de date de la senzorul local și afișăm aceste rezultate, după care trecem pe ce de-a doua linie a LCD-ului, și citim, respectiv afișăm datele ce vin pe legătura serială de la modulul XBee. Pentru a nu deveni o problemă vizuală pentru utilizator, datele sunt actualizate cu o întârziere de aproximativ 2 secunde.

CONCLUZII

Pot concluziona că în cadrul lucrării prezentate, am reușit să prezint toate caracteristicile majore ale protocoalelor de comunicație wireless folosite pentru transmisiile de date pe distanțe mici și mijlocii, scoțând în evidență avantajele lui ZigBee, atât din punct de vedere teoretic cât și practic, prin implementărilor realizate.

Cu ajutorul primului scenariu, în care am reglat intensitatea luminoasă a unui LED de la distanță, atât prin ajustarea unui potențiometru, cât și prin folosirea unui fotorezistor, am demonstrat faptul că sistemele ce înglobează ZigBee nu au neapărată nevoie de folosirea unui microcontroler intermediar sau a unei plăcuțe de dezvoltare. Modulele XBee conțin pini analogici ce pot citi date, punând acele date în rețea, pentru a fi citite și interpretate de către alte componente.

În cea de-a doua implementare folosită, prin implementarea unui sistem de sonerie wireless într-o clădire de apartamente, am dovedit scalabilitatea acestui protocol, care poate să fie folosit atât în cadre mici, cât și în unele mai extinse, în care să fie înglobate un număr mare de noduri, ce pot genera date într-un volum mare, în mod constant. Mai mult decât atât, serviciile ce pot fi conținute de o astfel de rețea pot să fie variate, ceea ce înseamnă ca în cadrul aceleași rețele putem avea și sistemul de sonerie dar și o mulțime de senzori care să acumuleze date, pe care să le trimită către un nod central.

Folosindu-mă de ultima topologie implementată am reușit să dovedesc faptul că ZigBee poate lucra cu o multitudine de noduri care să genereze în mod constant date, reușind să le afișeze pe toate fără a exista probleme în prelucrarea datelor. Acest lucru este posibil datorită modului în care se face multiplexarea datelor venite din mai multe locuri în același timp. Totuși, toate cele trei scenarii au ilustrat și posibilitatea folosirii acestui protocol pe distanțe destul de mari, întrucât în implementarea practică, semnalul a fost trimis cu succes inclusiv pe distanțe de aproximativ 35-40 de metri. Această

rază de acțiune este superioară altor protocoale wireless mult mai des folosite la ora actuală – Wi-Fi și Bluetooth.

Implementările realizate în cadrul acestei lucrări reprezintă un punct de început, întrucât numărul de aplicații care pot beneficia de pe urma acestui protocol este foarte mare, având deja optimizați mai mulți parametrii de funcționare – viteza de transmisiune, interferențele reduse între canale și alte dispozitive care funcționează în aceeași bandă de frecvență, dimensiunea variabilă a pachetelor de date transmise, consumul redus de energie al echipamentelor etc.

Toate scenariile implementate sunt utile într-un număr mare de locuri, atât în mediul casnic cât și în cel de birou, dar se poate merge și mai departe, cu cel de monitorizare al temperaturii, în folosirea într-un mediu de producție. Acest lucru este posibil datorită preciziei senzorului de temperatură folosit și datorită faptului că afișarea temperaturii este făcută în timp real, fără a exista întârzieri, nici măcar între variațiile de $\pm 0.1^{o}C$.

Mai departe mi-am propus includerea acestor proiecte și a altora care vor urma în cadrul unei locuințe ce să incorporeze mai multe concepte ce pot ilustra tehnologia Internet of Things. Acest lucru cuprinde inclusiv alcătuirea unor biblioteci pentru sistemele de operare mobile, astfel încât să existe posibilitatea interacționării cu protocolul ZigBee și dintr-un cadru mobil (smartphones și tablete). La momentul actual, existența acestor librării reprezintă singurul avantaj major deținut de către protocoalele Bluetooth și Wi-Fi, pe lângă scopul relativ mai scăzut. Scopul final este automatizarea proceselor ce se derulează în cadrul acestei locuințe, în vederea eficientizării proceselor ce se derulează în cadrul acesteia și a stilului de viață al oamenilor.

BIBLIOGRAFIE

[1] Perrilo, M., Heinzelman, W., „Wireless Sensor Network Protocols", Department of Electrica land Computer Engineering, 2015

[2] Faludi, R., „Building Wireless Sensor Networks", Editura O'Reilly, 2011

[3] Orner, R., Grünbacher, E., Guger, C., „State of the Art in Sensors, Signals and Signal Processing", în GmbH/Guger Technologies OG, 2009, pp 9 – 14.

[4] Lee, J., Suu Y., Shen, C., "A Comparative Study of Wireless Protocols"

Bluetooth, UWB, ZigBee, and Wi-Fi", The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taiwan, 2007

[5] Muqri, M., Alfaro, R., "BuildingWireless Sensor Networks with Zigbee", 12th ASEE Annual Conference and Exposition, 2013

[6] Bavarva, A., Pathel, N., "Wireless Sensors Networks using ZigBee", ResearchGate, 2016

[7] ZigBee Alliance, ZigBee Specification, Ianuarie 2008, ZigBee Document 05347r17

[8] Augusto, R., Severino, R., "On the use of IEEE 802.15.4/ZigBee for Time-Sensitive Wireless Sensor Network Applications", Polytechnic Institute of Porto

[9] XBee Datasheet

[10] DS18B20 Sensor Datasheet

[11] Olsson, J., Lönn, J., “ZigBee for wireless Networking”, Universitatea Linköping, lucrarea de disertație, 2015

[12] Bragisnky, D., Estrin, R., ”Rumor routing algorithm for sensor networks” – Editura O’Reilly, 2009

[13] http://www.zigbee.org - accesat la data: 11.06.2017

[14] https://www.digi.com/lp/xbee - accesat la data: 10.06.2017

[15] https://en.wikipedia.org/wiki/Zigbee - accesat la data: 12.06.2017

[16] https://www.arduino.cc/en/Reference/HomePage - accesat la data: 08.06.2017

[17] https://www.arduino.cc/en/Main/ArduinoBoardUno - accesat la data: 08.06.2017

ANEXA 1

Ministerul Educaţiei Naţionale

UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

DIPLOMĂ

la

Sesiunea de Comunicări Ştiinţifice Studenţeşti

MENŢIUNE

se acordă lucrării:

Comunicarea între noduri wireless cu protocolul *ZigBee*

realizata de

Horia Alexandru CĂLIN

anul IV A, Facultatea Electronică, Telecomunicaţii şi Tehnologia Informaţiei

sub coordonarea stiintifica a

Prof. dr. ing. Corneliu BURILEANU,

Departamentul Dispozitive, Circuite şi Aparate Electronice

PREŞEDINTELE SENATULUI,

Prof.dr.ing. Ecaterina ANDRONESCU

Mai 2017

DECAN,

Prof.dr.ing. Cristian NEGRESCU

ANEXA 2

```
const int potPin1 = A0;
int valoarePotentiometru1;
void setup() {
  Serial.begin(9600);
}
void loop() {
    //Se citesc valoarile de la porturile analogice si se stocheaza in variabile corespunzatoare
    valoarePotentiometru1 = analogRead(A0);
     //Mapam valoarile citite de la potentiometru la o valoare de tensiune PWM
    valoarePotentiometru1 = map (valoarePotentiometru1, 0, 1023, 0, 255);
   //Trimitem mesajul
  Serial.print('<');  //Simbol de start pentru transmisiune
```

```
  Serial.print(valoarePotentiometru1);//Valoarea citita va avea
valori intre 0 si 255
  delay(10);
  Serial.println('>');//Simbol pentru sfarsitul transmisiunii
}
```

ANEXA 3

```
const int ledRosu = 3;   //Vom lega led-urile la pini de tip PWM (Pulse-Wide-Modulation)


//Variables
bool transmisiuneInceputa= false; //Adevarat daca mesajul a fost inceput
bool transmisiuneIncheiata    = false;//Adevarat daca mesajul este incheiat
char byteCurent ;
char mesaj[3];
byte index;


void setup() {
  pinMode(ledRosu, OUTPUT);
     Serial.begin(9600);
}
```

```
void loop() {

  while (Serial.available()>0){
     //Se citeste byte-ul primit
    byteCurent = Serial.read();
   //Mesajul incepe cu simbolul '<'
    if(byteCurent == '<'){
     transmisiuneInceputa = true;
     index = 0;
     mesaj[index] = '\0'; // Se arunca orice pachete incomplete -
punem null in locul lor
   }
    //Mesajul se termina cu simbolul '>'
   else if(byteCurent == '>')
   {
     transmisiuneIncheiata = true;
     break;  //Iesirea din bucla dupa terminarea citirii
   }
   //Se citeste mesajul primit
   else
   {
     if(index < 4) // Ne asiguram ca vectorul nu este plin
     {
       mesaj[index] = byteCurent; // Add caracterul in vector
       index++;
       mesaj[index] = '\0'; // Adaugam null la final
     }
   }
 }
 if(transmisiuneInceputa && transmisiuneIncheiata){
   int valoareLedRosu = atoi(mesaj);
  analogWrite(ledRosu, valoareLedRosu);
   //Serial.println(value); //Folosit pentru depanare
   index = 0;
   mesaj[index] = '\0';
   transmisiuneInceputa = false;
   transmisiuneIncheiata = false;}
```

ANEXA 4

```
int luminaAmbientala = 11;
int troubleShootLED = 13;
int valoareAnalogica = 0;

void setup() {
  pinMode(luminaAmbientala,OUTPUT);
  pinMode(troubleShootLED,OUTPUT);
  Serial.begin(9600);

}

void loop() {
  if (Serial.available() >= 21) {
// Cautam byte-ul care semnaleaza inceputul transmisiei
    if (Serial.read() == 0x7E) {
      Serial.println("Citesc ce trebuie");
```

```
      //Pentru debug am folosit un LED ca sa confirmam primirea
datelor
      digitalWrite(troubleShootLED, HIGH);
      delay(10);
      digitalWrite(troubleShootLED, LOW);
      //citim variabilele pe care nu le vom folosi in afara
bufferului
      for (int i = 0; i<18; i++) {
        byte discard = Serial.read();
      }
      //memoram valorile de referinta
      int valoareAMare = Serial.read();
      int valoareAMica = Serial.read();
      valoareAnalogica = valoareAMica + (valoareAMare * 256);
    }
  }
  //in continuare mapam pe trei intervale aprinderea
  //scrierile seriale sunt folosite doar pentru troubleshoot
  if (valoareAnalogica > 0 && valoareAnalogica <= 350) {
    digitalWrite(luminaAmbientala, 0);
    Serial.println("Citesc valoare foarte mare - lumina mica");
   }
  if (valoareAnalogica > 350 && valoareAnalogica <= 750) {
      digitalWrite(luminaAmbientala, 128);
      Serial.println("Citesc valoare mare - lumina la jumatatea
intensitatii");
  }
  if (valoareAnalogica > 750 && valoareAnalogica <= 1023) {
      digitalWrite(luminaAmbientala, 255);
      Serial.println("Citesc valoare mica - lumina mare");
  }
}
```

ANEXA 5

```
const int butonEtaj1 = 13;
const int butonEtaj2 = 10;
const int ledRosu = 8;
const int ledGalben = 7;
const int ledVerde = 9;
const char a = 'A';
const char b = 'B';
const String s1 = "Se asteapta comanda";
const String s2 = "Proiect wireless Zigbee";
const String s3 = "A fost apasat butonul 1";
const String s4 = "A fost apasat butonul 2";
const String s5 = "Se initializeaza conexiunea";
/*
 * Initializarea variabilelor programului
 */
```

```
int initializareZigbee = 0;

/*
 * Se stabilesc rolurile pinilor si se initializeaza ecranul LCD
 */
void setup() {
  pinMode(butonEtaj1, INPUT);
  pinMode(butonEtaj2, INPUT);
  pinMode(ledRosu, OUTPUT);
  pinMode(ledGalben, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  lcd.begin(16, 2);
  lcd.print(s2);
  Serial.begin(9600);
}

void loop() {
  lcd.setCursor(0, 1);

  if(initializareZigbee == 0){
      lcd.print(s5);
      initializareZigbee++;
     delay(15000);
    }
  /*
   * Daca nu este apasat niciun buton, se va afisa un mesaj corespunzator pe LCD si clipeste, la interval de 0.5 secunde, un LED verde
   *
   */
  if((digitalRead(butonEtaj1) == LOW) && (digitalRead(butonEtaj2) == LOW)){
      digitalWrite(ledVerde, HIGH);
      delay(100);
      digitalWrite(ledVerde, LOW);
      delay(100);
      lcd.print(s1);
      //Serial.println(s1); ***pentru debugging
```

```
    }
    /*
     * Daca se apasa primul buton, se afiseaza un mesaj corespunzator
pe LCD si se aprinde constant ledul rosu
     */

  if (digitalRead(butonEtaj1) == HIGH) {
    Serial.println(a);
    lcd.print(s3);
    digitalWrite(ledRosu, HIGH);
    //delay(50);
    digitalWrite(ledRosu, LOW);
  }

  /*
     * Daca se apasa al doilea buton, se afiseaza un mesaj
corespunzator pe LCD si se aprinde constant ledul galben
     */

  if(digitalRead(butonEtaj2) == HIGH){
    Serial.println(b);
    lcd.print(s4);
    digitalWrite(ledGalben, HIGH);
    delay(50);
    digitalWrite(ledGalben, LOW);
    }
}
```

ANEXA 6

```
const int sonerieEtaj1 = 5;
const int ledVerde = 6;
int sunat = 0;

void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(sonerieEtaj1, OUTPUT);
  Serial.begin(9600);
}
void loop() {
if (Serial.available() > 0) {
    if (Serial.read() == 'A'){
    Serial.println("Primesc semnal");
    digitalWrite(sonerieEtaj1, HIGH);
    tone(sonerieEtaj1, 1000, 500);
    digitalWrite(ledVerde, HIGH);
```

```
    delay(100);
    //digitalWrite(sonerieEtaj1, LOW);
    digitalWrite(ledVerde, LOW);
    }
  }
}
```

ANEXA 7

```
#include <LiquidCrystal.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define pinSenzorTemperatura 7

LiquidCrystal displayLCD(12, 11, 5, 4, 3, 2);
OneWire oneWire(pinSenzorTemperatura);
DallasTemperature senzorTemperatura(&oneWire);

String  afisajSenzor1;

void setup() {
  Serial.begin(9600);
```

```
  displayLCD.begin(16, 2);
  senzorTemperatura.begin();

}
void loop(){
displayLCD.setCursor(0,0);
    afisajSenzor1 = Serial.readString();
    if ((afisajSenzor1.equals("\n")) == false){
         Serial.println(afisajSenzor1);
         displayLCD.print(afisajSenzor1);
         delay(1000);
    }
      String afisajSenzor2 = "Senzor 2: ";
      sensors.requestTemperatures();
      String citire1 = (String)sensors.getTempCByIndex(0);
      afisajSenzor2 += citire1;
      afisajSenzor2 += "C";
      Serial.println(t);
      displayLCD.setCursor(0,1);
      displayLCD.print(afisajSenzor2);
      delay(1000);
}
```

ANEXA 8

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define pinSenzorTemperatura 2
OneWire oneWire(pinSenzorTemperatura);
DallasTemperature senzorTemperatura(&oneWire);
void setup(void){
  Serial.begin(9600);
  senzorTemperatura.begin();}
void loop(void){
  String afisajSenzor = "Senzor 1: ";
  senzorTemperatura.requestTemperatures();
  String valoareSenzor = (String)sensors.getTempCByIndex(0);
  afisajSenzor += valoareSenzor;
  afisajSenzor += "C";
  Serial.println(afisajSenzor);
  delay(1000);
}
```