Universitatea POLITEHNICA București
Facultatea Automatică și Calculatoare
Departamentul Automatică și Informatică Industrială

LUCRARE DE LICENȚĂ

Integrarea unei plăci de dezvoltare bazate pe microcontrolerul ATSAMR21 cu mediul de dezvoltare Arduino IDE

Blânzeanu Doru-Florin

Coordonator științific:

Sl. dr. ing. Radu Pietraru

BUCUREȘTI

2018

Cuprins

1 Introducere

În lucrarea de față se va vorbi despre integrarea unei noi plăci de dezvoltare, Atmel SAMR21 Xplained Pro, în ecosistemul Arduino. Deasemenea, se va face o analiză asupra domeniului Internetului Lucrurilor(IoT)[1] în care se utilizează microcontrolerul acestei placi de dezvoltare și a importanței circuitelor care îmbină puterea de calcul cu comunicația radio.

Pentru dezvoltarea de sisteme integrate exista opțiunea de a alege componenta de bază pentru procesare fie un microcontroler, fie un microprocesor. Ambele abordări au avantajele și dezavantajele lor, dar în general în aplicațiile în care contează dimensiunea, costul și puterea consumată, este preferat un microcontroler. Când vine vorba despre o aplicație care necesită comunicație radio, alegerea unui microcontroler care are integrată o componentă radio este mult mai convenabilă spre deosebire de un microcontroler care nu dispune de această componentă la care să se adauge componenta radio externă. Unul dintre motivele pentru care este mai convenabilă prima variantă este constrângerea asupra dimensiunii, care în cazul primei opțiuni, este mult mai mică. Un alt motiv pentru care reprezintă un avantaj alegerea unui microcontroler cu modul radio integrat, este faptul că producătorii acestor microcontrolere au ca scop maximizarea performanței prin minimizarea consumului, astfel aceste microcontrolere sunt specializate în comunicația radio spre deosebire de celelalte care sunt de uz general.

Arduino este o platformă open-source bazată pe componente hardware și unelte software ușor de utilizat. Ecosistemul Arduino oferă suport și pentru o gamă largă de alte microcontrolere, toate aceste unelte alăturate într-un singur pachet ușor de utilizat pentru dezvoltatorii începători, însă destul de flexibil pentru dezvoltatorii avansați. Arduino este foarte utilizat in diverse domenii cum ar fi:

- robotică
- internetul lucrurilor
- învățământ
- prototipare

[1] IoT = Internetul of Things

- cercetare și dezvoltare

Datorită faptului că Arduino este o platformă open-source, oricine este binevenit să contribuie la dezvoltarea de noi funcționalități și integrarea de noi plăci de dezvoltare. Acest lucru are și avantajul că există mulți utilizatori care lucrează pe același cod sursă, astfel totodată involuntar testând și raportând orice tip de problemă.

În ultimii ani, Arduino a atras din ce în ce mai mulți utilizatori dornici să implementeze aplicații practice rapid deoarece arhitectura platformei are grijă să inițializeze și să configureze multe dintre elementele utilizate, astfel este mult mai simplu pentru utilizatorii mai puțin experimentați să înceapă un proiect nou, iar utilizatorii cu experiență pot opta să configureze manual doar elementele sau modulele de care ei au nevoie.

2 Prezentarea domeniului din care face parte lucrarea

Internetul Lucrurilor se referă la un tip de rețea care conectează totul la internet printr-o suită de protocoale în scopul schimbării de informație și a comunicării pentru a putea implementa recunoaștere, poziționare, urmărire și administrare inteligentă[1].
Internetul lucrurilor promite să revoluționeze felul în care noi trăim și muncim. Ne-ar putea ajuta să trecem peste problemele globale datorate populației: criza energiei, lipsa resurselor și poluare. Pentru a realiza această viziune, avem nevoie de senzori care preiau informația din mediu și o împărtășesc între ei sau cu alte dispozitive care permit accesul utilizatorului uman pentru a lua decizii inteligente care afecteaza întregul nostru ecosistem. Datorită acestui fapt, interesul către internetul lucrurilor este din ce în ce mai mare. Multe studii au prevăzut o creștere accelerată a numărului de vânzări de dispozitive inteligente în următorii 10 ani.

Imaginați-vă o lume în care milioane de obiecte, dispozitive are putea comunica și împărtăși informația între ele, toate interconectate printr-o rețea internet. Toate datele colectate de acele obiecte(senzori) fiind centralizate în anumite centre de colectare a datelor care au o interfață de expunere către utilizator.
Aplicațiile care se pot dezvolta ulterior să folosească acele date pot ajuta omenirea în diverse domenii: industrie, sănătate, stil de viață, comunicații, etc. Ne apropiem cu pași repezi de acel ideal, însă suntem deocamdată departe deoarece există încă destule probleme care apar din cauza multor factori precum:

- cost
- securitate
- performanță
- consum

Potrivit unui studiu efectuat în anul 2018 cu privire la clasificarea domeniilor în care activează proiecte din domeniul internetului lucrurilor spune că majoritatea proiectelor sunt în sectorul de orașe inteligente, industrie și clădiri inteligente[2]. Pe continentul american se află majoritatea acestor proiecte, urmat de Europa și Asia. Există diferențe mari între regiuni și

domeniile de activitate ale proiectelor, astfel majoritatea proiectelor pentru orașe inteligente se află pe teritoriul european în timp ce pe continentul american predomină proiecte de mașini inteligente și sănătate. Asia este puternic dezvoltată în domeniul agriculturii inteligente.

Dorința de a funcționa un timp îndelungat fără să necesite intervenția omului, a fost scopul principal în designul de rețele de senzori wireless în ultimii 10 ani[5]. Prin concentrarea asupra acestei cerințe, designerii au căutat acele componente hardware care aveau cel mai mic consum de curent electric în mod activ și în sleep mode[1]. Aceste dispozitive consumă energie de ordin mW putere în mod activ, iar în modul de consum redus uW, renunțând la putere de calcul și memorie în favoarea consumului redus. În schimb, puterea de calcul limitată și resursele de memorie mici ale acestor platforme restricționează aplicațiile pe care le pot implementa. Aplicațiile tipice urmăresc un model care preia date de la senzori, îi stochează, îi trimite și apoi intră din nou în modul de consum redus, în care senzorii de pe placă sunt interogați și datele returnate sunt trimise către un server. Aplicațiile ce necesită putere de calcul mare nu sunt suportate de aceste platforme.
Aplicațiile care necesită putere de calcul mare sau procesare de semnal, în general trebuie să facă un compromis pentru a evita limitările impuse de platformă. Aceste aplicații tind să fie caracterizate de intervale alternante de activitate intensă urmată de perioade de inactivitate. Alternativa este să se utilizeze platforme care sacrifică consumul în favoarea puterii de procesare, sau platforme care combină microcontrolere low-power[2] cu procesoare de înaltă performanță care consumă mult.

2.1 Aplicații

Internetul lucrurilor se poate aplica în activitățile industriale, inclusiv în tranzacțiile dintre companii, organizații sau alte entități, se poate utiliza în logistică, producție, procese, servicii, banking, etc[3].

[1] sleep mode = mod de consum redus
[2] low-power = consum redus

2.1.1 Industrie

Logistică și managementul ciclului de viață al produsului

În logistică și managementul ciclului de viață al produsului un exemplu bun de utilizare a internetului lucrurilor este în procesul de selecție a produselor. Există etichete electronice atașate pe anumite obiecte care pot fi utilizate în identificarea tipului de material: îmbrăcăminte, mobilier, echipamente, alimente sau băuturi. Utilizarea etichetelor electronice contribuie la gestionarea eficientă a spațiului de depozitare și reduce inventarul. Întregul proces poate fi urmărit, spre exemplu utilizând un cititor de carduri RFID[3] instalat la o fabrică care poate monitoriza procesul de producție și fiecărei etichete i se poate vedea întreg parcursul în fabrică. Un sistem avansat compus din echipament RFID și urmărire în timp real a produselor de pe rafturi poate ajuta în reducerea deșeurilor, astfel reducând din costuri.

Agricultură și creșterea animalelor

În acest domeniu, internetul lucrurilor, contribuie prin urmărirea în timp real a poziției animalelor pentru a putea raporta în caz de anumite evenimente, cum ar fi boli, către autorități în timp util. Sistemele IoT de identificare permit monitorizarea și identificarea animalelor și izolează animalele infectate de animalele sănătoase, astfel evitând răspândirea bolilor infecțioase. Există sisteme avansate care pot memora informații despre condițiile fizice ale animalelor și le pot transmite astfel facilitând analiza datelor colectate pe care autoritățile le pot verifica.
Un alt exemplu bun din agricultură sunt fermele inteligente care au devenit visul oricărui fermier, să poți supraveghea cultura, să poți acționa de la distanță în consecință și să poți monitoriza starea solului în timp real sunt doar câteva dintre aplicațiile pe care le oferă IoT.

Procese industriale

În industria automobilelor, domeniul IoT are o vastă aplicabilitate, de la senzori de monitorizare a parametrilor funcționării automobilului, presiune în pneuri, consum de carburant, poziție, distanțe față de alte vehicule, până la automatizarea proceselor din industrie și a

[3]RFID = Radio-frequency identification

liniilor de fabricație. Datele de la senzorii amplasați în sistem pot oferi detalii importante care ușurează identificarea problemelor și rezolvarea acestora.

2.1.2 Oraș inteligent

IoT va îmbunătăți susținerea mediului și calitatea vieții oamenilor. Principala resursă este energia și modul eficient de utilizare al acesteia, precum și soluții inteligente pentru îmbunătățirea vieții de zi cu zi a oamenilor.

Clădiri inteligente

Integrarea tehnologiei de comunicație în clădiri inteligente permite ca viitoarele clădiri sau orașe inteligente să fie echipate cu o varietate de senzori și dispozitive inteligente interconectate precum:

- telefoane mobile
- computer
- TV
- camere de supraveghere
- electrocasnice inteligente

Unele aplicații permit funcții de bază IoT cum ar fi: sisteme de supraveghere, sistem de gestionare și mentenanță a fabricilor, sisteme multimedia; în timp ce alte aplicații integrează smart grid[4] și optimizează consumul energiei. Spre exemplu, HAN[5] permite electrocasnicelor să interacționeze cu instrumente inteligente, și asigură performanța cerută reducând costul. Poate deasemnea să programeze aceste electrocasnice să nu funcționeze în perioada de vârf. Toate aceste informații sunt accesibile utilizatorilor prin intermediul unei aplicații mobile.

[4]smart grid = electrical grid which includes a variety of operational and energy measures
[5]HAN = Home Area Network

Figura 1: Interconectarea sistemelor inteligente din clădiri

Siguranță publică și monitorizarea mediului

Siguranța publică include menținerea ordinii publice, protecția cetățenilor și protecția de proprietate publică și privată. IoT oferă soluții de monitorizare și urmărire a situațiilor de urgență. Sistemele de urgență ajută la prevenirea și răspunsul în consecință în cazul unui cataclism. Prin colecționarea datelor de la camerele de luat vederi publice sau private se poate ajuta poliția să păstreze liniștea publică. Clădirile care necesită o securitate mai ridicată pot beneficia de sisteme IoT pentru protecție și detecție împotriva intrușilor. Senzori dedicați și camere inteligente, sistemul GPS[6] care oferă date despre poziționare, locație în timp real, precum și utilizarea tehnologiei wireless[7] pot ajuta la anticiparea anumitor evenimente.

2.1.3 Sănătate

Internetul lucrurilor va juca un rol important în dezvoltarea de servicii inteligente pentru îmbunătățirea activității sociale a oamenilor. IoT care implică cetățeni și comunități în guvernare și luare de decizii pentru a permite oamenilor să trăiască independent sau să mențină relații sociale și să îmbunătățească sănătatea.

[6]GPS = Global Positioning System

[7]wireless = fără fir

Diagnosticarea bolilor și tratament

Industria sănătății va fi puternic influențată de IoT. Senzori inteligenți care permit colectarea de informații vitale de la pacienți(temperatură, tensiune arterială, puls, nivel de colesterol) în timp real și transmise către un specialist printr-o tehnică de comunicație pentru diagnostic și monitorizare. Rețele BAN[8] sunt interconectate prin dispozitive purtabile care permit monitorizare de la distanță a pacienților din afara spitalelor. Deasemenea IoT mai poate fi utilizat în identificarea substanțelor, ustensilelor în spitale și inventarierea acestora, astfel pierderea sau furtul acestora fiind mai puțin probabile.

Stil de viață

În cadrul acestui subdomeniu, IoT are o largă întrebuințare. Oamenii pot opta să utilizeze dispozitive purtabile care includ senzori de monitorizare a semnelor vitale care pot declanșa alarme medicale în cazul unor probleme de sănătate, precum și ale altor indici de interes pentru utilizator(calorii consumate, distanță parcursă, poziționare). Activitățile zilnice pot fi și acestea urmărite și se pot primi sugestii de schimbare a unor obiceiuri în funcție de scopul setat de utilizator. Pentru persoanele cu handicap vizual sunt dezvoltate aplicații IoT(senzori și aplicații mobile) care le ajută pe acestea să se deplaseze în oraș.

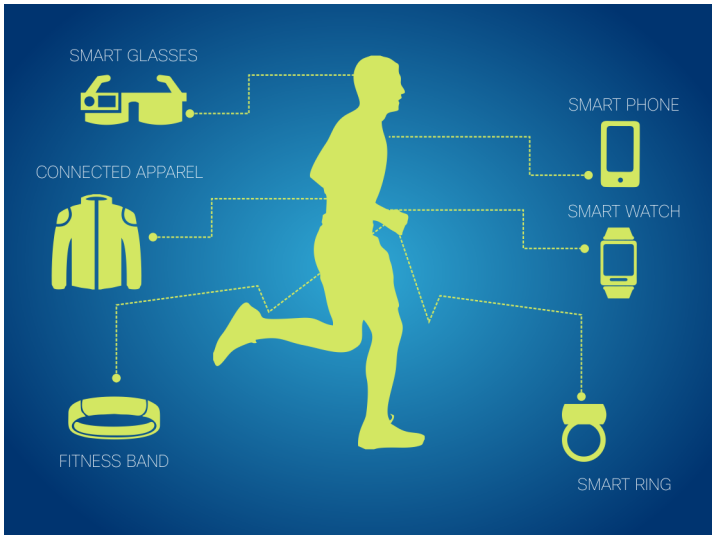

Figura 2: Exemplu de sisteme ce ajută la îmbunătățirea stilului de viață

[8]BAN = Body Area Network

3 Descrierea problemei abordate și a metodei de rezolvare propuse

În cadrul lucrării se abordează problematica integrării unei noi plăci de dezvoltare în ecosistemul Arduino.
Gama SAMR21 de la Atmel cuprinde o serie de microcontrolere cu consum redus ce pot fi utilizate în aplicații care necesită comunicație radio. Kitul de dezvoltare Atmel SAM R21 Xplained Pro este o platformă hardware proiectată de Atmel pentru a se putea evalua microcontrolerul ATSAMR21G18A.
Motivele pentru care un dezvoltator de aplicații IoT ar alege o placă de dezvoltare bazată pe microcontrolerul ATSAMR21G18A sunt:

- consumul redus
- performanța ridicată
- modulul radio integrat

Când vine vorba de alegerea unui mediu de dezvoltare prietenos cu utilizatorii începători și care suportă o gamă largă de platforme și plăci de dezvoltare, opțiunile sunt limitate, cea mai des aleasă metodă este utilizarea Arduino IDE. Arduino IDE permite utilizatorilor să dezvolte aplicații care folosesc plăci de dezvoltare bazate pe microcontrolere: AVR, SAMD, STM32, Intel ix86, etc. După cum se observă, plăcile de dezvoltare bazate pe microcontrolere SAMR, nu sunt suportate.
Tema propusă este să se adauge, la lista de plăci de dezvoltare suportate de către Arduino IDE, și placa Atmel SAMR21 Xplained Pro.

Primul pas în rezolvarea acestei probleme este de a analiza structura proiectului unei alte plăci de dezvoltare și a urma acel model. După analiza proiectului unei plăci de dezvoltare bazate pe SAMD s-a ajuns la concluzia că pentru adăuga o placă de dezvoltare la Arduino IDE trebuie să adăugăm o nouă componentă nucleu care să fie compatibilă cu placa de dezvoltare Atmel SAMR21 Xplained Pro și fișiere de configurare pentru pinii plăcii.

4 Documentație tehnică

În capitolul curent se vor prezenta detaliile tehnice folosite în rezolvarea temei propuse.

4.1 Placa Atmel SAMR21 Xplained Pro

Pentru a putea duce la bun sfârșit tema propusă, au fost necesare două plăci de dezvoltare bazate pe microcontrolerul ATSAMR21G18A, două plăci Atmel SAM R21 Xplained Pro (Figura 3).
Placa de dezvoltare Atmel SAM R21 Xplained Pro este o platformă hardware de evaluare a microcontrolerului ATSAMR21G18A, produs de Atmel[6].

Figura 3: Atmel SAMR21 Xplained Pro

Cele mai importante caracteristici ale plăcii de dezvoltare sunt:

- procesor ARM Cortex-M0+ de până la 48MHz

- memorie flash 256KB
- memorie SRAM 32KB
- mod de consum redus
- depanator integrat(EDBG)
    - interfață USB[1]
    - programarea și depanarea prin SWD[2]
- pini de intrare și ieșire
- două butoane mecanice
- un LED
- antenă radio
    - antenă ceramică[8]
    - un conector SMA[3] pentru antenă externă
- trei posibilități de alimentare
    - extern
    - prin depanator USB
    - prin USB țintă
- modulu radio AT86RF233[9]
- cristal de 32kHz
- cristal de 16MHz

4.1.1 Procesor

Procesorul ARM Cortex-M0+ are un set de 56 de instrucțiuni, majoritatea pe 16 biți, și operează pe regiștri de 32 de biți fiind proiectat cu scopul de a servi în aplicații cu sisteme integrate[12]. Oferă beneficii semnificative dezvoltatorilor, inclusiv:

- arhitectură simplu de învățat și utilizat
- mod de consum ultra redus
- procesare de întreruperi de mare performanță
- compatibil cu procesoare Cortex-M

[1]USB = Universal Serial Bus
[2]SWD = Serial Wire Debug
[3]SMA = SubMiniature version A

- include sistem de protecție a memoriei

Toți regiștrii sunt pe 32 de biți dintre care 13 regiștri sunt de uz general, 3 sunt pentru uz special(SP[4], LR[5], PC[6]). Procesorul ARM Cortex M0+ poate adresa un maxim de 4GB de memorie care sunt împărțiți în memorie pentru cod, SRAM, periferice, RAM extern, dizpozitive externe, magistrala privată a perifericelor și memorie sistem precum în figura 4.

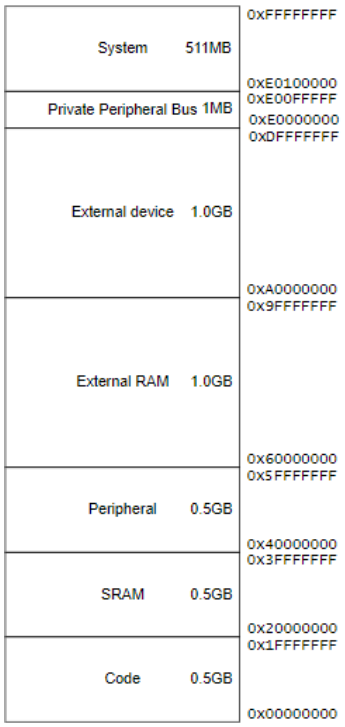

Figura 4: Modelul structurării memoriei[13]

Procesorul Cortex-M0+ este construit pe un nucleu optimizat pentru consum și dimensiune, cu un pipeline pe două niveluri(Figura 5) și o arhitectură von Neumann.

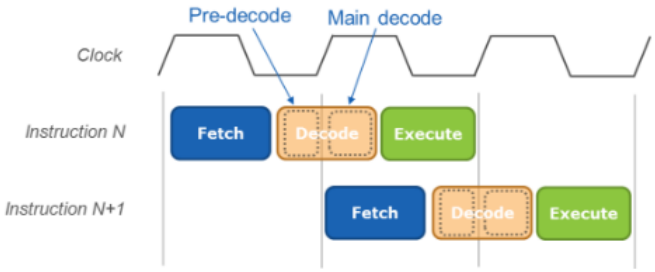

Figura 5: Pipeline pe două niveluri[14]

[4]SP = Stack Pointer
[5]LR = Link Register
[6]PC = Program Counter

Acest procesor implementează arhitectura ARMv6-M, care e bazată pe un set de instrucțiuni de 16 biți Thumb și include tehnologie Thumb-2. Acest lucru oferă performanțele excepționale așteptate de la un procesor cu arhitectură pe 32 de biți, cu o densitate mai mare a codului decât microcontrolerele de 8 sau 16 biți.

Procesorul ARM Cortex M0+ dispune de un controler pentru întreruperi, NVIC[7], care se ocupă cu gestionarea acestora. NVIC permite prioritizarea întreruperilor, mascarea aceastora și definirea de rutine care se apelează la declanșarea unei întreruperi. Executarea rutinelor se face dupa 15 sau 16 cicli de ceas, în acest timp, procesorul gestionează trecerea de la stiva funcției executate la momentul de timp actul la stiva pentru rutina de întrerupere.

4.1.2 Depanatorul integrat

Placa Atmel SAM R21 Xplained Pro conține depanatorul Atmel EDBG[8] pentru depanarea codului sursă direct pe placa de dezvoltare. EDBG este compus din trei interfețe: un depanator, un port de comunicație serială COM[9] și o interfață de date DGI[10].
Depanatorul este utilizat pentru depanarea codului prin rularea lui pe placa de dezvoltare și verificarea bunei funcționări.
Portul de comunicație serială este conectat la o interfață UART[11], interfață de comunicație serială asincronă, și oferă posibilitatea de comunicare cu microcontrolerul prin terminal software. Oferă viteză de transfer ridicată, opțiuni de paritate a datelor transmise și de bit de stop.
Interfața de date e compusă din mai multe interfețe de comunicație cu computerul gazdă. Comunicația pe aceste interfețe este bidirecțională și poate fi folosită pentru notificarea în legătură cu declanșarea unor evenimente sau pur și simplu pentru transmisie de date.

4.1.3 Surse de alimentare

Placa de dezvoltare Atmel SAM R21 Xplained Pro poate fi alimentată în mai multe moduri, după cum arată tabela 1.

[7]NVIC = Nested Vectored Interrupt Controller
[8]EDBG = Atmel Embedded Debugger
[9]COM = Communication Port
[10]DGI = Data Gateway Interface
[11]UART = Universal Asynchronous receiver-transmitter

Placa de dezvoltare va detecta automat ce surse de alimentare sunt disponibile și va alege dintre acestea în funcție de următoarele priorități:

1. Extern
2. Prin depanator USB
3. Prin USB țintă

Tabela 1: Surse alimentare

Intrare	**Tensiune**	**Curent**
Extern	5V ±100 mV pentru utilizare ca gazdă. De la 4.3V până la 5.5V dacă nu e necesară utilizarea ca gazdă	Minimul recomandat e 1A pentru a putea să susțină toate dispozitivele conectate. Recomandat este maxim 2A.
Prin depanator USB	De la 4.4V până la 5.25V	500mA
Prin USB țintă	De la 4.4V până la 5.25V	500mA

4.1.4 Periferice

Butoane mecanice

Placa de dezvoltare conține doua butoane mecanice. Un buton este butonul de reset care este conectat ma linia de reset a microcontrolerului, iar celălalt buton este un buton de uz general care poate fi configurat. Când unul dintre butoane este apăsat, va lega linia la GND[12].

LED

Există un LED, numit LED0, disponibil pe placa de dezvoltare Atmel SAM R21 Xplained Pro care poate fi activat sau dezactivat. Pentru a activa ledul, este necesar să legăm linia la GND.

[12]GND = ground (masă)

Radio

Principalul scop al plăcii de dezvoltare Atmel SAMR21 Xplained Pro, este să pună în evidență capacitățile de comunicație radio ale microcontrolerului ATSAMR21G18A. Modulul AT86RF233, este un transmițător radio de consum redus proiectat pentru uz industrial și pentru aplicații ce folosesc protocoalele IEEE 802.15.4, Zigbee, SP100, WirelessHART, ISM. Este un periferic SPI-to-antenna, adică folosește o comunicație SPI pentru a permite interacțiunea dintre microcontroler și antenă. Modulul este compus dintr-un transmițător radio analogic și un demodulator digital care include sincronizare pe domeniile timp și frecvență. Toate componentele necesare sunt integrate într-un singur chip, astfel minimizând numărul de componente externe: antena, cristal de quartz și capacități de decuplare.

Această placă dispune de posibilitatea de a alege dintre două antene, antena ceramică sau antena conectată la conectorul SMA. Această posibilitate o oferă intrerupătorul AS222-92LF[10] conectat la doi pini de la microcontroler, RFCTRL1 și RFCTRL2, prin care microcontrolerul alege care antenă va fi utilizată în procesul de comunicație. Alegerea se face pe baza ieșirii unui transformator 2450BM15A0015[11] care este conectat la 2 pini diferențiali de antenă de la microcontroler ca în figura 6.

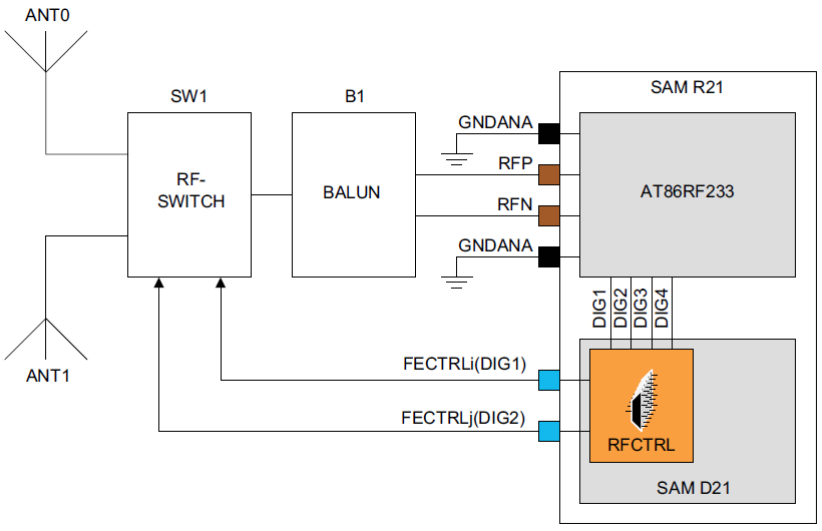

Figura 6: Schema de legare a antenei

4.2 Arduino IDE

Înainte de a trece mai departe, este necesară prezentarea structurii unui proiect din Arduino IDE și detalii despre cum se ajunge la obiectul final încărcat pe placa de dezvoltare țintă. Un proiect în Arduino IDE este împărțit în 4 părți cu cod sursă:

- core[13] - aici este codul de bază pentru fiecare platformă
- variant[14] - aici se află fișiere de configurare pentru placa de dezvoltare
- sketch[15] - aici se află codul utilizatorului
- library[16] - aici se află codul sursă pentru bibliotecile opționale

Punctul de intrare în proiect este fișierul *main.c* în funcția *main*, din *core*, în care se apelează funcțiile scrise de utilizator în fișierul cu extensia *.ino*, *setup și loop*, din *sketch*.
Procesul de build[17] al proiecului constă în compilarea tuturor fișierelor sursă, cu fișierele header corespunzătoare, din toate cele 4 părți ale proiectului și asamblarea lor într-un singur fișier rezultat cu extensia *.hex*, stocat la o locație temporară. Fișierul rezultat este mai apoi urcat în memoria microcontrolerului țintă și executat.

4.2.1 Dezvoltare proiect

Pentru a implementa tema propusă, se vor creea 3 din cele 4 părți componente ale unui proietct în Arduino IDE. În final, cele 3 părți formează pachetul care va fi arhivat și stocat online, iar oricine va dori să utilizeze o placă de dezvoltare de acest tip, va putea folosi acest pachet. La baza acestui pachet se află câteva fișiere de configurare a programatoarelor folosite de această placă de dezvoltare, de configurare a plăcii de dezvoltare și de configurare a procesului de build.

Fișierul *boards.txt* conține informații esențiale cu privire la placa de dezvoltare cum ar fi:

- name - numele plăcii de dezvoltare

[13] core = nucleu
[14] variant = se referă la placa de dezvoltare
[15] sketch = schiță
[16] library = bibliotecă
[17] build = construire

- vid - număr de identificare a vânzătorului
- pid - număr de identificare a producătorului
- upload tool - unealta folosită pentru a urca codul pe placa de dezvoltare precum și parametri de configurare a acesteia
- mcu - procesorul pentru care se face build
- core - nucleul utilizat
- ldscript - linker script
- variant - variantul folosit

Fișierul folosit în cadrul proiectului conține toate acele informații, dar și alte informații suplimentare necesare funcționării corecte a programării plăcii de dezvoltare(Figura 7).

```
Atmel_SAMR21_XplainedPro_edbg.name=Atmel SAMR21 Xplained Pro No Bootloader(via EDBG)
Atmel_SAMR21_XplainedPro_edbg.vid.0=0x03eb
Atmel_SAMR21_XplainedPro_edbg.pid.0=0x2111
Atmel_SAMR21_XplainedPro_edbg.build.vid=0x2341
Atmel_SAMR21_XplainedPro_edbg.build.pid=0x804d
Atmel_SAMR21_XplainedPro_edbg.upload.tool=openocd
Atmel_SAMR21_XplainedPro_edbg.upload.protocol=swd
Atmel_SAMR21_XplainedPro_edbg.upload.maximum_size=262144
Atmel_SAMR21_XplainedPro_edbg.upload.use_1200bps_touch=false
Atmel_SAMR21_XplainedPro_edbg.upload.wait_for_upload_port=false
Atmel_SAMR21_XplainedPro_edbg.upload.native_usb=false
Atmel_SAMR21_XplainedPro_edbg.build.mcu=cortex-m0plus
Atmel_SAMR21_XplainedPro_edbg.build.f_cpu=48000000L
Atmel_SAMR21_XplainedPro_edbg.build.usb_product="Atmel SAMR21-XPRO"
Atmel_SAMR21_XplainedPro_edbg.build.usb_manufacturer="Atmel Corp"
Atmel_SAMR21_XplainedPro_edbg.build.board=SAMD_ATMEL_SAMR21_XPRO
Atmel_SAMR21_XplainedPro_edbg.build.core=arduino
Atmel_SAMR21_XplainedPro_edbg.build.extra_flags=-D__SAMR21G18A__ {build.usb_flags}
Atmel_SAMR21_XplainedPro_edbg.build.ldscript=linker_scripts/gcc/flash_without_bootloader.ld
Atmel_SAMR21_XplainedPro_edbg.build.openocdscript=openocd_scripts/variant_atmel_edbg.cfg
Atmel_SAMR21_XplainedPro_edbg.build.variant=atmel_samr21_xpro
Atmel_SAMR21_XplainedPro_edbg.build.variant_system_lib=
```

Figura 7: Fișierul *boards.txt* asociat pachetului

Pentru configurarea programatorului folosit este nevoie de un fișier în care se specifică numele programatorului, care se va găsi în meniul cu programatoare din Arduino IDE, și tipul de comunicație folosit , în cazul proiectului de față fiind USB. Mai este necesar un fișier numit *platform.txt* în care se specifică argumente pasate compilatorului, linkerului sau asamblorului.

Core

Ținând cont de faptul că placa de dezvoltare Atmel SAM R21 Xplained Pro este un dispozitiv SAM R21, care este compus din SAM D21 și modulul radio AT86RF233, codul sursă pentru core va fi asemănător cu cel pentru SAM D21, cu modificările aferente.
După o examinare amănunțită a documentației microcontrolerului SAM D21G în comparație cu cea a microcontrolerului SAM R21G, se observă câteva diferențe majore:

- numărul de module TC[18] a scăzut de la 5 la 3
- pentru SAM R21G nu avem modul DAC[19]
- pentru SAM R21G nu avem modul I2S[20]

Pentru a ne putea folosi de codul deja scris pentru plăcile de dezvoltare bazate pe SAM D21, este necesară modificarea acestuia în conformitate cu documentația oferită de producător pentru SAM R21G. Configurarea corectă a acestuia implică modificarea numărului de module TC și eliminarea inițializării modulului DAC, pe care microcontrolerele SAM D21 le au. Fiecare întrerupere are asociată o rutină care se execută când este declanșată acea întrerupere. Aceste rutine sunt setate în în mod implicit în codul de Arduino să apeleze funcție numită *Dummy_Handler* care va bloca programul în cazul unei întreruperi nedorite de utilizator. La setarea unei întreruperi prin apelul la functia *attachInterrupt*, se modifică vectorul de întreruperi, variabila *exception_table*(Figura 8), în care sunt stocate adresele rutinelor de întrerupere pentru fiecare dintre întreruperi.
Această variabilă, este pusă implicit în secțiunea de cod numită *.isr_vector* folosind **__attribute__((section()))**. Astfel, în fișierul binar rezultat, această variabilă va fi pusă în memorie unde indică scriptul pentru linker. Pentru a configura corect întreruperile pentru placa de dezvoltare Atmel SAM R21 Xplained Pro, trebuie să scoatem definirea rutinelor pentru modulele DAC, I2S și TC. Astvel, se vor înlătura *TC6_Handler, TC7_Handler, DAC_Handler, I2S_Handler*.

La declanșarea întreruperii de reset, fie prin apăsarea butonului sau software, se inițializează datele din RAM cu valorile aflate în zona flash, zona de date neinițializate se setează cu 0, iar apoi se apelează funcția de inițializare a modulelor care activează ceasul pentru toate modulele, pornește modulele esențiale. După ce inițializarea s-a terminat cu succes, se apelează funcția

[18] TC = Timer Counter
[19] DAC = Digital to Analog Converter
[20] I2S = Inter-IC Sound Interface

```
/* Configurable interrupts */
(void*) PM_Handler,             /*  0 Power Manager */
(void*) SYSCTRL_Handler,        /*  1 System Control */
(void*) WDT_Handler,            /*  2 Watchdog Timer */
(void*) RTC_Handler,            /*  3 Real-Time Counter */
(void*) EIC_Handler,            /*  4 External Interrupt Controller */
(void*) NVMCTRL_Handler,        /*  5 Non-Volatile Memory Controller */
(void*) DMAC_Handler,           /*  6 Direct Memory Access Controller */
(void*) USB_Handler,            /*  7 Universal Serial Bus */
(void*) EVSYS_Handler,          /*  8 Event System Interface */
(void*) SERCOM0_Handler,        /*  9 Serial Communication Interface 0 */
(void*) SERCOM1_Handler,        /* 10 Serial Communication Interface 1 */
(void*) SERCOM2_Handler,        /* 11 Serial Communication Interface 2 */
(void*) SERCOM3_Handler,        /* 12 Serial Communication Interface 3 */
(void*) SERCOM4_Handler,        /* 13 Serial Communication Interface 4 */
(void*) SERCOM5_Handler,        /* 14 Serial Communication Interface 5 */
(void*) TCC0_Handler,           /* 15 Timer Counter Control 0 */
(void*) TCC1_Handler,           /* 16 Timer Counter Control 1 */
(void*) TCC2_Handler,           /* 17 Timer Counter Control 2 */
(void*) TC3_Handler,            /* 18 Basic Timer Counter 0 */
(void*) TC4_Handler,            /* 19 Basic Timer Counter 1 */
(void*) TC5_Handler,            /* 20 Basic Timer Counter 2 */
(void*) (0UL),                  /* 21 Reserved */
(void*) (0UL),                  /* 22 Reserved */
(void*) ADC_Handler,            /* 23 Analog Digital Converter */
(void*) AC_Handler,             /* 24 Analog Comparators */
(void*) (0UL),                  /* 25 Reserved */
(void*) PTC_Handler,            /* 26 Peripheral Touch Controller */
(void*) (0UL),                  /* 27 Reserved */
```

Figura 8: Întreruperile configurabile din *exception_table*

main, care va apela mai departe codul din sketch, al utilizatorului.

Este necesară și îndepărtarea codului de inițializare a modulului DAC, deoarece pentru microcontrolerul SAM D21G se pornea ceas și pentru modulul de DAC, iar la microcontrolerul SAM R21G nu există, setarea acelui bit poate duce la un comportament nedefinit. În final, este necesară doar activarea modulului ADC în PM[21] setând bitul ADC din registrul APBCMASK(Figura 9).

0x20	APBCMASK	7:0	SERCOM5	SERCOM4	SERCOM3	SERCOM2	SERCOM1	SERCOM0	EVSYS	PAC2
0x21		15:8			TC5	TC4	TC3	TCC2	TCC1	TCC0
0x22		23:16			RFCTRL		PTC		AC	ADC
0x23		31:24								

Figura 9: Formatul registrului APBCMASK

În fișierul *WVariant.h*, în enumul *EPioType*, mai definim un element numit *PIO_RADIO* care va avea valoarea 5. Această modificare va ajuta ulterior la definirea pinilor radio pentru placa de dezvoltare, care au nevoie de funcția F(Figura 10).

[21] PM = Power Manager

Internal Signal	I/O Pin	Supply	Type	A	B				C	D	E	F	G	H
				EIC	REF	ADC	AC	PTC	SERCOM	SERCOM-ALT	TC	FECTRL TCC SERCOM	COM	AC/ GCLK
CLKM	PC16	VDDIO	I/O									GCLK/ IO[1][1]		
SCLK	PC18	VDDIO	I/O									SERCOM4/ PAD[3]		
MISO	PC19	VDDIO	I/O									SERCOM4/ PAD[0]		

Figura 10: Funcțiile alternative ale pinilor radio

Variant

În această parte a proiectului sunt definite configurațiile pinilor plăcii de dezvoltare, precum și diferite scripturi pentru linker. În fișierul *variant.cpp*, pinii GPIO[22] sunt descriși printr-o structură numită *PinDescription*(Figura 11).

```
typedef struct _PinDescription
{
  EPortType       ulPort ;
  uint32_t        ulPin ;
  EPioType        ulPinType ;
  uint32_t        ulPinAttribute ;
  EAnalogChannel  ulADCChannelNumber ;
  EPWMChannel     ulPWMChannel ;
  ETCChannel      ulTCChannel ;
  EExt_Interrupts ulExtInt ;
} PinDescription ;
```

Figura 11: Structura folosită pentru definirea pinilor GPIO

EPortType reprezintă portul de care aparține pinul respectiv, acesta putând fi: PORTA, PORTB, PORTC, iar următorul membru al structurii conține numărul pinului. Membrul *ulPinType* conține informații despre funcția periferică alternativă pe care o îndeplinește pinul, iar următoarele câmpuri sunt utilizate pentru setarea de atribute, intreruperi, canal pentru ADC.

Fișierul conține un vector de structuri de acest tip care reprezintă pinii pe care îi poate folosi un utilizator când lucrează cu această placă de dezvoltare. Fișierul header mai conține și niște definiții pentru anumiți pini care vor fi utilizați ulterior în cod, spre exemplu pinii pentru modulul radio.

Scripturile pentru linker reprezintă un set de reguli pe care linkerul le urmează când procesul

[22]GPIO = General Purpose I/O

de build ajunge în faza de linkare. Aceste reguli se referă la locul unde sunt stocate variabilele, codul și simbolurile globale. Acest script pentru linker este format din două părți:

- Memory - în care se declară memoriile disponibile, adresele de început și dimensiunile acestora
- Sections - în care se atribuie fiecare secțiune de memorie din obiectele compilate la una din memoriile disponibile

Un exemplu de secvență în care se definesc memoriile disponibile se poate observa în Figura 12, unde sunt definite două memorii, FLASH și RAM fiecare cu dimensiunile respective.

```
MEMORY
{
  FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 0x00040000
  RAM (rwx) : ORIGIN = 0x20000000, LENGTH = 0x00008000
}
```

Figura 12: Memoriile definite în scriptul pentru linker

În segmentul *Sections* se pot rula diferite comenzi, însă cele mai utilizate în cadrul acestui proiect sunt acelea de a selecta secțiuni, de a le atribui la una din memoriile definite și de a defini simboluri care pot fi folosite în cod pe parcursul dezvoltării. Un exemplu este zona de date *.bss*(Figura 13) care conține toate variabilele globale neinițializate încadrate între două simboluri globale *__bss_start__*, *__bss_end__* folosite în general la determinarea dimensiunii.

```
.bss :
{
    . = ALIGN(4);
    __bss_start__ = .;
    *(.bss*)
    *(COMMON)
    . = ALIGN(4);
    __bss_end__ = .;
} > RAM
```

Figura 13: Zona de memorie .bss din scriptul pentru linker

Library

Codul sursă pentru funcționalitatea radio se află în această parte a proiectului. Implementarea pentru biblioteca radio care oferă funcționalitatea dorită folosind modulului AT86RF233

există în sistemul de operare în timp real destinat aplicațiilor IoT, RIOT-OS. Translatarea acestui cod la o bibliotecă scrisă în C++ pentru Arduino va fi folosită în realizarea proiectului[15].

Pentru a putea utiliza modulul radio AT86RF233 este necesară configurarea corectă a acestuia. Motivul pentru care configurarea a fost implementată aici și nu în inițializarea plăcii de dezvoltare este că activarea acestui modul radio duce la un consum adițional care va exista doar când este folosită această bibliotecă. În fișierul sursă al clasei *AT86RF2xx* am adăugat niște metode adiționale care vor ajuta la configurarea corectă a microcontrolerului pentru a comunica cu modulul radio prin intermediul unei interfețe SPI[23](Figura 14).

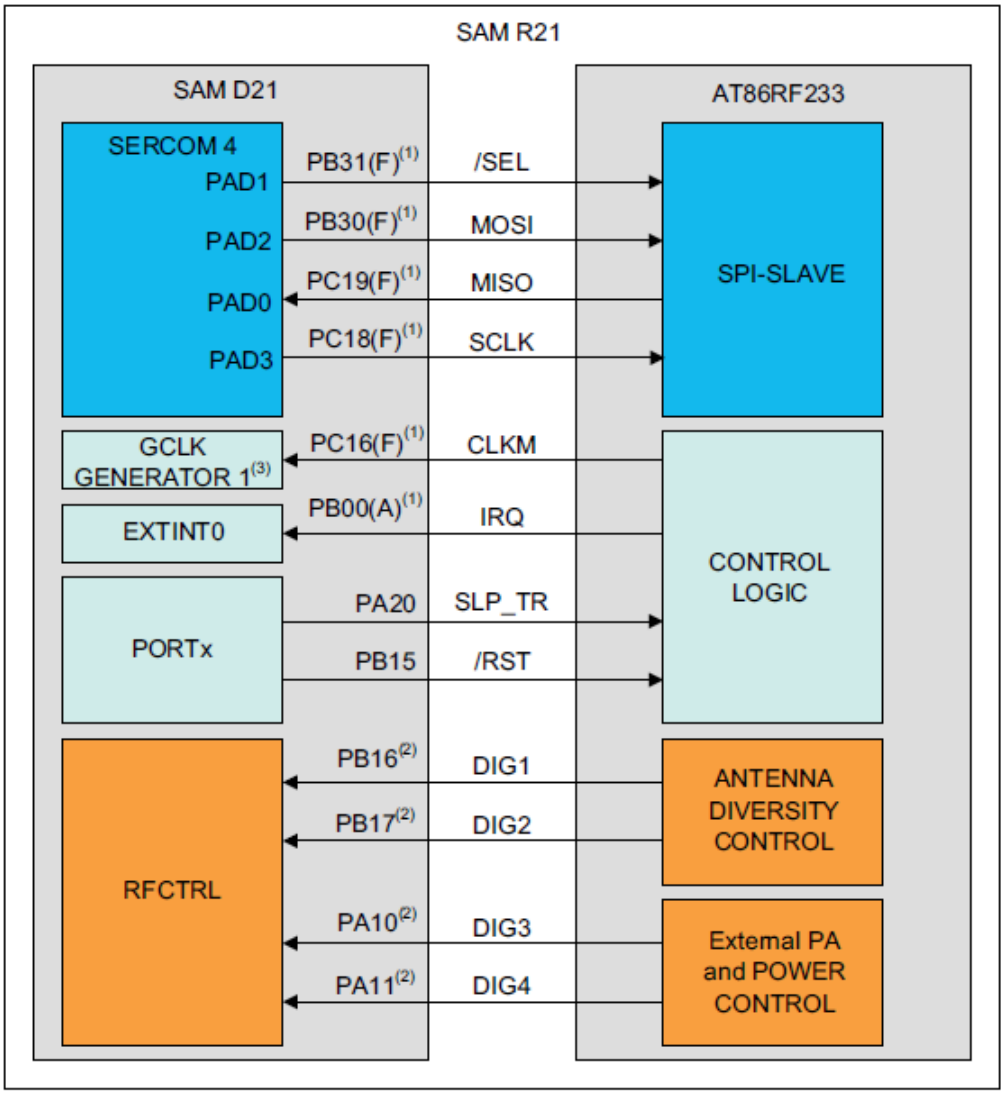

Figura 14: Interfața de comunicație dintre microcontreler și modulul radio[6]

[23]SPI = Serial Peripheral Interface

Etapele configurării sunt:

1. Setare pin /SEL ca ieșire și setarea ieșirii acestuia pe High
2. Activare Sercom4
3. Setarea ceasului corect și pornirea acestuia pentru Sercom4[24]
4. Setare pini MISO, MOSI, SCK ca intrare cu pulldown
5. Pornire comunicație SPI folosind biblioteca SPI din Arduino
6. Configurare întrerupere

Pentru setarea pinilor ne folosim de faptul că am configurat corect intrările din *variant* și putem astfel să apelăm rutinele din Arduino și să fim siguri că se întâmplă ceea ce dorim. La apelul funcției *pinMode* se setează în registrul DIR din PORT direcția dorită(High sau Low). Pentru a putea folosi pinii pe comunicația SPI cu modulul radio, intern biblioteca SPI setează niște funcții periferice alternative, și anume funcția F, în registrul PMUX din PORT și activează multiplexorul din registrul PINCFG folosindu-se de registrul WRCONFIG, care ajută la configurarea unui grup de pini(Figura 15).

```
if ( g_APinDescription[ulPin].ulPin & 1 ) // is pin odd?
{
  uint32_t temp ;

  // Get whole current setup for both odd and even pins and remove odd one
  temp = (PORT->Group[g_APinDescription[ulPin].ulPort].PMUX[
      g_APinDescription[ulPin].ulPin >> 1].reg) & PORT_PMUX_PMUXE( 0xF ) ;
  // Set new muxing
  PORT->Group[g_APinDescription[ulPin].ulPort].PMUX[g_APinDescription[
      ulPin].ulPin >> 1].reg = temp|PORT_PMUX_PMUXO( ulPeripheral ) ;
  // Enable port mux
  PORT->Group[g_APinDescription[ulPin].ulPort].PINCFG[g_APinDescription[
      ulPin].ulPin].reg |= PORT_PINCFG_PMUXEN ;
}
else // even pin
{
  uint32_t temp ;

  temp = (PORT->Group[g_APinDescription[ulPin].ulPort].PMUX[
      g_APinDescription[ulPin].ulPin >> 1].reg) & PORT_PMUX_PMUXO( 0xF ) ;
  PORT->Group[g_APinDescription[ulPin].ulPort].PMUX[g_APinDescription[
      ulPin].ulPin >> 1].reg = temp|PORT_PMUX_PMUXE( ulPeripheral ) ;
  PORT->Group[g_APinDescription[ulPin].ulPort].PINCFG[g_APinDescription[
      ulPin].ulPin].reg |= PORT_PINCFG_PMUXEN ; // Enable port mux
}
```

Figura 15: Pașii urmați pentru setarea funcției periferice

Deoarece registrii PMUX[0-15], în care sunt stocate funcțiile periferice alternative pentru pini

[24]Sercom = Serial Communication Interface

sunt împărțiți în două părți egale, una pentru pinul cu număr par, iar cealaltă pentru pinul cu număr impar, este necesară acea verificare pentru a ști să prezervăm setările deja făcute pentru pinul care folosește același registru, dar de paritate diferită.
Pentru activarea Sercom4 se setează bitul corespunzător din registrul APBCMASK din PM. Configurarea ceasului pentru Sercom4 se face folosing regiștrii din GCLK[25]. Astfel, primul pas este să se dezactiveze canalul de ceas pentru Sercom4_core, urmată de selectarea ceasului dorit prin scrierea în registrul CLKCTRL din GCLK a numărului de identificare corespunzător și a sursei de ceas dorite(în cazul nostru 0, ceasul principal), iar apoi activarea ceasului. Aceiași pași se urmează și la activarea Sercom4_slow. Sercom4_core este necesar când operează ca master, iar Sercom4_slow este necesar pentru diferite funcții. Pentru setarea întreruperii este necesară structura pinului corespunzător să fie configurată corect, iar rutinele din Arduino vor seta corect și întreruperea pentru modulul radio. Pinul pentru întreruperea venită de la modulul radio, IRQ, va fi setat ca pin de intrare cu pull down iar pentru atașarea unei rutine de întrerupere se va apela funcția *attachInterrupt*, care va seta pe întreruperea externă 0, EXTINT0, o întrerupere pe ciclul crescător al semnalului.

După ce configurarea s-a terminat, biblioteca radio verifică dacă poate citi cu succes registrul *PART_NUM* din modulul radio, folosind comunicația SPI.
Pentru a comunica cu modulul radio prin intemediul SPI s-au definit mai multe moduri de acces:

- Register access citire și scriere
- Frame Buffer access citire și scriere
- SRAM access citire și scriere

Pentru a putea selecta unul dintre cele 6 moduri disponibile pentru comunicația cu modulul AT86RF233, se transmite un octet care va coda comanda după cum arată Figura 16.
Modul *Register Access* este folosit pentru a scrie sau a citi regiștri modulului AT86RF233, adresele acestora se află în intervalul 0x00-0x3F. Acest mod de acces reprezintă o transmisie de doi octeți, operație declanșată de setarea bitului de selecție /SEL pe LOW. Primul octet este format din doi biți care reprezintă comanda transmisă(citire sau scriere), urmată de 6 biți în care se stochează adresa registrului accesat, iar cel de al doilea octet reprezintă valoarea care se dorește a fi scrisă în acel registru în cazul operației de scriere, în cazul operației

[25]GCLK = Generic Clock Controller

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Access Mode	Access Type
1	0	Register address [5:0]						Register access	Read access
1	1	Register address [5:0]							Write access
0	0	1	Reserved					Frame Buffer access	Read access
0	1	1	Reserved						Write access
0	0	0	Reserved					SRAM access	Read access
0	1	0	Reserved						Write access

Figura 16: Descrierea comenzilor transmise pe SPI[7]

de citire nu contează. Protocolul SPI este un protocol de transmisie dublă, adică în timpul tranferului de date de la microcontroler la modulul radio se face și un transfer invers care în cazul citirii este valoarea cerută, iar în cazul operației de scriere este o valoare care reflectă starea modulului radio.

În biblioteca radio implementată, acest mod de acces este utilizat pentru citira stărilor modulului radio și pentru începerea transmisiei de date. Cel mai des utilizați regiștri sunt: TRX_STATE, TRX_CTRL_0, TRX_CTRL_2 și IRQ_MASK[7].

Frame Buffer este o zonă de memorie SRAM de 128 de octeți cu două porturi. Un port este conectat la interfața SPI, celălalt la transmițătorul și receptorul intern. Ambele porturi sunt independente și accesibile simultan. Acest buffer utilizează spațiul de adrese SRAM, 0x00-0x7F, pentru operații de transmisie și recepție și poate stoca un singur cadru IEEE 802.15.4 de lungime maximă la același moment de timp. Un cadru IEEE 802.15.4 este format din două părți, o secțiune de preambul generată intern și partea din *Frame buffer*, accesibilă utilizatorului. În partea de *Frame buffer* se reține și date despre lungimea cadrului, puterea sau calitatea acestuia(LQI[26]), o estimare a puterii semnalului primit(ED[27]), etc(Figura 17).

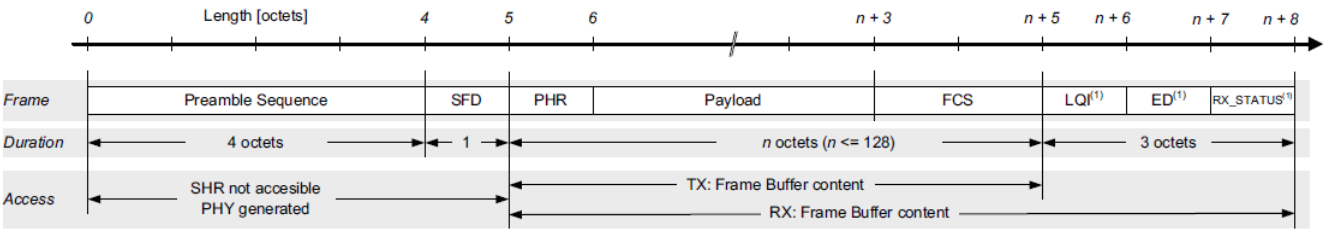

Figura 17: Formatul unui cadru IEEE 802.15.4[7]

Modul de acces *Frame buffer* este utilizat când se dorește citirea sau scrierea unui cadru IEEE

[26]LQI = Link Quality Indication
[27]ED = Energy Detection

802.15.4. Adresa de scriere sau citire este întotdeauna resetată la 0 și incrementată, astfel pentru a putea citi sau scrie informațiiile adiționale despre cadre.

Spre deosebire de modul de acces *Frame buffer*, modul de acces SRAM, este utilizat pentru a scrie în bufferul pentru cadru începând de la o adresă specificată. Oferă posibilitatea accesului direct la datele utile de la adresa dorită fără să se mai urmeze procesul de incrementare pe care îl abordează accesul *Frame Buffer*.
În biblioteca radio, modul de acces *Frame Buffer* este folosit doar pentru citirea dimensiunii unui cadru primit, iar în rest, pentru transmisia cadrului se folosește modul de acces SRAM. Astfel, pentru transmiterea unui șir de octeți se apelează următoarele metode din cadrul clasei AT86RF2XX:

- tx_prepare
- tx_load
- tx_exec

În cadrul funcției *tx_prepare* se verifică starea modulului, dacă este ocupat se așteaptă eliberarea acestuia, iar dacă primește un cadru se anulează transmisia. Se contunuă cu setarea stării de așteptare transmitere. Funcția de *tx_load* are ca parametri șirul de octeți care vor fi transmiși, lungimea șirului și indicele de unde va începe scrierea. Datorită faptului că valoarea câmpului PHR este de un octet, indicele de unde va începe scrierea va fi egal cu 1. Pentru a scrie în memoria SRAM se apelează funcția de transfer din clasa SPI care va pune în registrul de date octetul care se dorește a fi transmis spre modulul radio. Astfel, datele transmise sunt, comanda care selectează modul de acces, indicele de la care se începe scrierea și un număr de octeți egal cu lungimea șirului transmis. Pentru a porni transmisia, în metoda *tx_exec*, se scrie dimensiunea cadrului IEEE 802.15.4 care va urma a fi transmis, aceasta va fi egală cu dimensiunea șirului de octeți la care se mai adaugă un octet pentru dimensiunea cadrului, și se va scrie în registrul de stare, comanda de pornire a transmisiei. Toată această procedură poate fi observată în Figura 18.

```
size_t AT86RF2XX::send(uint8_t *data, size_t len)
{
    /* check data length */
    if (len > AT86RF2XX_MAX_PKT_LENGTH) {
        Serial.println("[at86rf2xx] Error: Data to send exceeds max packet size.");
        return 0;
    }
    AT86RF2XX::tx_prepare();
    AT86RF2XX::tx_load(data, len, 0);
    AT86RF2XX::tx_exec();
    return len;
}

void AT86RF2XX::tx_prepare()
{
    uint8_t state;

    /* make sure ongoing transmissions are finished */
    do {
        state = get_status();
    }
    while (state == AT86RF2XX_STATE_BUSY_TX_ARET);

    /* if receiving cancel */
    if(state == AT86RF2XX_STATE_BUSY_RX_AACK) {
        force_trx_off();
        idle_state = AT86RF2XX_STATE_RX_AACK_ON;
    } else if (state != AT86RF2XX_STATE_TX_ARET_ON) {
        idle_state = state;
    }
    set_state(AT86RF2XX_STATE_TX_ARET_ON);
    frame_len = IEEE802154_FCS_LEN;
}

size_t AT86RF2XX::tx_load(uint8_t *data,
                          size_t len, size_t offset)
{
    frame_len += (uint8_t)len;
    sram_write(offset + 1, data, len);
    return offset + len;
}

void AT86RF2XX::tx_exec()
{
    /* write frame length field in FIFO */
    sram_write(0, &(frame_len), 1);
    /* trigger sending of pre-loaded frame */
    reg_write(AT86RF2XX_REG__TRX_STATE, AT86RF2XX_TRX_STATE__TX_START);
}
```

Figura 18: Implementarea transmisieu unui șir de octeți

4.2.2 Adăugare proiect

Pentru adăugarea unei noi plăci de dezvoltare la mediul de dezvoltare Arduino IDE, este necesară crearea unui fișier de tip json[28] în care se scriu configurațiile pentru pachetul corespunzător plăcii de dezvoltare. Fișierul json trebuie să conțină următoarele câmpuri:

- name - numele pe care îl va avea pachetul
- maintainer - numele celui care menține codul
- websiteURL - un url către un site web al companiei

[28] JSON = JavaScript Object Notation

- help
- platforms

    name - numele plăcii de dezvoltare

    arhitecturre - arhitectura pe care se bazează placa de dezvoltare

    version - versiunea pachetului

    category - de obicei este Contributed, doar cei de la Arduino pot schimba

    url - link către o arhivă cu codul sursă

    archiveFilename - numele arhivei

    checksum - pentru verificare se cere un mesaj de autentificare

    size - mărimea arhivei

    boards

    toolsDependencies - secțiunea cu dependențele de alte utilitare
- tools - aici se specifică explicit de unde se pot luat utilitarele

Un exemplu de fișier de configurare se poate vedea în figura 19.

```
"platforms":
[
  {
    "name": "Atmel SAMR21 Xplained-pro",
    "architecture": "samr",
    "version": "0.1.0",
    "category": "Contributed",
    "url": "https://github.com/blanzeanudoru/lucrare_licenta/releases/download/v0.1.0/SAMR21_board-0.1.0.tar.bz2",
    "archiveFileName": "SAMR21_board-0.1.0.tar.bz2",
    "checksum": "SHA-256:dd3437d2e5397437f654e79ece75ac8ce6d91a0dd5a260e7ba9f9c46598709fa",
    "size": "826149",
    "boards": [
      {
        "name": "atmel_samr21_xpro"
      }
    ],
    "toolsDependencies":
    [
      {
        "packager": "atmel-samr21-xpro",
        "name": "arm-none-eabi-gcc",
        "version": "4.8.3-2014q1"
      },
      {
        "packager": "atmel-samr21-xpro",
        "name": "bossac",
        "version": "1.6.1-arduino"
      },
      {
        "packager": "atmel-samr21-xpro",
        "name": "openocd",
        "version": "0.9.0-arduino6-static"
      },
      {
        "packager": "atmel-samr21-xpro",
        "name": "CMSIS",
        "version": "4.5.0"
      },
      {
        "packager": "atmel-samr21-xpro",
        "name": "CMSIS-Atmel",
        "version": "1.2.0"
      }
    ]
  }
],
```


Figura 19: Exemplu de fișier de configurare

4.3 Rezultate obținute

5 Concluzii și dezvoltări ulterioare

6 Bibliografie

[1] Keyur K Patel, Sunil M Patel. *Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges*. ISSN 2321 3361, 2016 IJESC
http://ijesc.org/upload/8e9af2eca2e1119b895544fd60c3b857.Internet%20of%20Things-IOT%20Definition,%20Characteristics,%20Architecture,%20Enabling%20Technologies,%20Application%
accesat la data 10/06/2018.

[2] Padraig Scully. *The Top 10 IoT Segments in 2018 – based on 1,600 real IoT projects*. 2018 ranking of Top IoT Segments
https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/ accesat la data 10/06/2018.

[3] Ran Liu, Jinfeng Wang. *Internet of Things: Application and Prospect* MATEC Web of Conferences 100, 02034 (2017)
https://www.matec-conferences.org/articles/matecconf/pdf/2017/14/matecconf_gcmm2017_02034.pdf
accesat la data 10/06/2018.

[4] Sayfe Kiaei, Eby G. Friedman. *Introduction to the Special Issue on Low Power Wireless Communications*. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING, VOL. 44, NO. 6, JUNE 1997.
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=868450 accesat la data 10/06/2018.

[5] JeongGil Ko, Kevin Klues, Christian Richter, Wanja Hofer, Branislav Kusy, Michael Bruenig, Thomas Schmid, Qiang Wang, Prabal Dutta, and Andreas Terzis *Low Power or High Performance? A Tradeoff Whose Time Has Come (and Nearly Gone)*
https://pdfs.semanticscholar.org/1f68/63c5c7c5e22a4507f1993f6c4c4a8b0b5594.pdf accesat la data 10/06/2018.

[6] *Atmel-42243D-SAM-R21-Xplained-Pro_User Guide-04/2016*

[7] *Atmel-42223G–SAM-R21_Datasheet–05/2016*

[8] 2450AT18D0100 datasheet. *2.45 GHz SMD Antenna, EIA 1210, Detuning resilient, Edge Mount Design P/N 2450AT18D0100*. Johanson Technology.

[9] AT86RF233 datasheet. *Low Power, 2.4GHz Transceiver for ZigBee, RF4CE, IEEE 802.15.4, 6LoWPAN, and ISM Applications*.

[10] *AS222-92, AS222-92LF: PHEMT GaAs IC SPDT Switch 0.1–3 GHz datasheet*.

[11] *2.45GHz Impedance Matched Balun-Filter for Atmel Chipset AT86RF232 and AT86RF233 datasheet*.

[12] *Cortex-M0+ Devices*. Generic User Guide. http://infocenter.arm.com/help/topic/com.arm.doc.dui0662b

[13] *Cortex M0+ memory map.*. Cortex-M0+ Devices Generic User Guide. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0662b/CHDBIJJE.html accesat la data 17/06/2018.

[14] *ARM Cortex®-M0+ Pipeline.*. Microchip Developer Help. http://microchipdeveloper.com/32arm:m0-pipeline accesat la data 17/06/2018.

[15] *Mark Solters*. Simple Arduino driver for the AT86RF233 802.15.4 radio module, ported to C++ from RIOT-OS. https://github.com/msolters/arduino-at86rf233 accesat la data 10/06/2018.