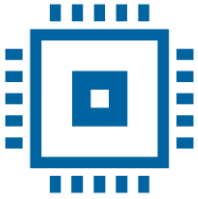

Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

PROIECT DE DIPLOMĂ

Conducător științific:
Șef Lucr. Dr. Ing. CROITORU Otilia

Absolvent:
ANGHEL Monica - Mihaela

BRAȘOV, 2024

Departamentul de Electronică și Calculatoare
Programul de studii: Tehnologii și Sisteme de Telecomunicații

ANGHEL Monica - Mihaela

Sistem de cartografiere spații inaccesibile și transmisie prin LoRa

Conducător științific:
Șef Lucr. Dr. Ing. CROITORU Otilia

Brașov 2024

Cuprins

LISTA DE FIGURI, TABELE ȘI CODURI SURSĂ

FIGURI

TABELE

CODURI SURSĂ

LISTA DE ACRONIME

IoT – Internet of Things
LoRa – Long Range
CSS – Chirp Spread Spectrum
ISM – Industrial, Scientific and Medical
FSK – Frequency Shift Keying
PA – Power Amplifier
BT – Bandwidth Time
FHSS – Frequency Hopping Spread Spectrum
MAC – Media Acces Control
LPWAN – Low Power Wide Area Network
CSMA-CA – Carrier Sense Multiple Access with Collision Avoidance
CPU – Central Processing Unit
DSP - Digital Signal Processor
UART – Universal Asynchronus Receiver-Transmitter
IIC – I^2C Bus Interface
SPI – Serial Peripheral Interface
CAN – Controller Area Network
PCM – Pulse Code Modulation
A/D - Analog/Digital
D/A – Digital/Analog
PWM – Pulse Width Modulation
LCD – Liquid Crystal Display
CMOS – Complementaru Metal Oxid Semiconductor
QFP - Quad Flat Package
MIPS - Million Instruction Per Second
ICSP - In Circuit Serial Programming
RF - Radio Frequency
GPIO - General Purpose Input/Output
CRC - Cyclic Redundancy Check
FIFO - First In, First Out
MLF – Micro Lead Frame
DMA – Direct Memory Access

1 INTRODUCERE

1.1 ASPECTE GENERALE

În era modernă, tehnologia a devenit omniprezentă, influențând profund aproape fiecare aspect al vieții noastre. Dispozitivele inteligente și comunicațiile digitale sunt fundamentale pentru funcționarea eficientă a infrastructurilor, a industriilor și a societății în ansamblu. Tehnologii avansate, precum Internetul Lucrurilor (IoT), rețelele de comunicații fără fir și sistemele de achiziții de date, contribuie semnificativ la optimizarea proceselor, automatizarea sarcinilor și crearea de noi oportunități de inovare și dezvoltare.

Achiziția de date reprezintă un proces esențial în multe domenii tehnice, permițând colectarea și analiza informațiilor din mediul înconjurător pentru a monitoriza, controla și optimiza diverse procese și sisteme.

Senzorii sunt dispozitive esențiale în sistemele de achiziție de date, capabile să detecteze și să măsoare parametrii fizici sau chimici și să transforme aceste măsurători în semnale electrice interpretabile de microcontrolere sau alte dispozitive de procesare. Utilizarea senzorilor în aplicațiile IoT este crucială pentru monitorizarea variabilelor de mediu.

Comunicațiile fără fir sunt fundamentale pentru transferul de date între dispozitive fără necesitatea utilizării cablurilor fizice, folosind unde radio sau alte forme de transmisie electromagnetică. Aceste tehnologii sunt esențiale pentru sistemele IoT, unde este necesară comunicarea eficientă între dispozitive repartizate geografic. [1]

LoRa este o tehnologie de comunicații fără fir, optimizată pentru aplicații IoT care necesită transmisii de date pe distanțe lungi, cu un consum redus de energie. LoRa utilizează tehnici de modulație cu spectru împrăștiat pentru a asigura o comunicație robustă și fiabilă, chiar și în medii zgomotoase sau aglomerate.

Prelucrarea informațiilor reprezintă un aspect crucial în sistemele de achiziție de date, implicând conversia datelor colectate de senzori într-o formă utilizabilă și interpretabilă. Acest

proces include filtrarea, analiza și vizualizarea datelor, asigurându-se că informațiile relevante sunt extrase și prezentate în mod clar și concis.

Tehnologiile de achiziție de date, prelucrarea informațiilor și comunicațiile fără fir reprezintă bazele pe care se dezvoltă inovațiile contemporane.

1.2 TEMA PROIECTULUI

Proiectul intitulat "Sistem de cartografiere spații inaccesibile și transmisie prin LoRa " se concentrează pe dezvoltarea unei mașinuțe autonome echipată cu senzori și module de comunicații fără fir LoRa, având scopul de a cartografia o peșteră. Această inițiativă combină atât aspecte hardware cât și pe cele software, oferind o soluție integrată și complexă pentru explorarea și documentarea mediilor inaccesibile.

Mașinuța va fi echipată cu senzori cu ultrasunete care vor măsura distanțele față de pereții peșterii. Senzorii sunt conectați la un microcontroler Arduino UNO, care va gestiona achiziția și transmiterea datelor.

Comunicarea între mașinuță și laptop se va realiza prin intermediul a două module LoRa RFM95. Un modul LoRa de pe mașinuță va transmite datele către un modul similar conectat la un al doilea Arduino la intrarea în peșteră. Astfel un modul LoRa va funcționa ca transmițător, montat pe mașinuță, în timp ce celălalt va funcționa ca receptor, conectat la un laptop pentru a prelua datele și a le prelucra.

Datele recepționate vor fi stocate într-o bază de date InfluxDB, ceea ce va permite stocarea și interogarea eficientă a acestora. Vizualizarea datelor se va realiza direct în InfluxDB, oferind reprezentări clare și detaliate ale configurației spațiului cartografiat.

Se utilizează Arduino IDE pentru a programa microcontrolerele, gestionând senzorii și modulele de comunicație. Codul scris în C++ configurează senzorii, gestionează comunicațiile și asigură transmiterea datelor către laptop. Pe laptop, un script Python preia și prelucrează informațiile recepționate, stocându-le eficient în baza de date InfluxDB pentru vizualizare și analiză ulterioară.

Proiectul demonstrează cum tehnologiile moderne pot fi aplicate în explorarea și cartografierea mediilor inacceesibile.

1.3 STRUCTURA PROIECTULUI

1.3.1 Componente

Acest proiect se împarte în două component principale: hardware și software.

A. Interfața cu utilizatorul

1. Componente hardware
   - Laptop/PC

2. Componente software
   - Arduino IDE
   - InfluxDB

B. Sistemul

1. Componente hardware
   - Arduino UNO R4 Minima
   - Arduino UNO R3
   - Module LoRa - RFM95
   - Senzor cu ultrasunete – HC-SR04
   - Rezistori
   - Sursă de alimentare – Baterie 9V
   - Cablu USB
   - Placă adaptoare pentru module – ESP8266
   - Breadboard
   - Conectori mamă-tată
   - Soclu baterie

2. Componente software
   - Limbaj de programare C++
   - Limbaj de programare Python

1.3.2 Schema de principiu a proiectului

În Figura 1 este prezentată schema de principiu a sistemului în care sunt prezentate toate componentele hardware care sunt folosite în realizarea proiectului.

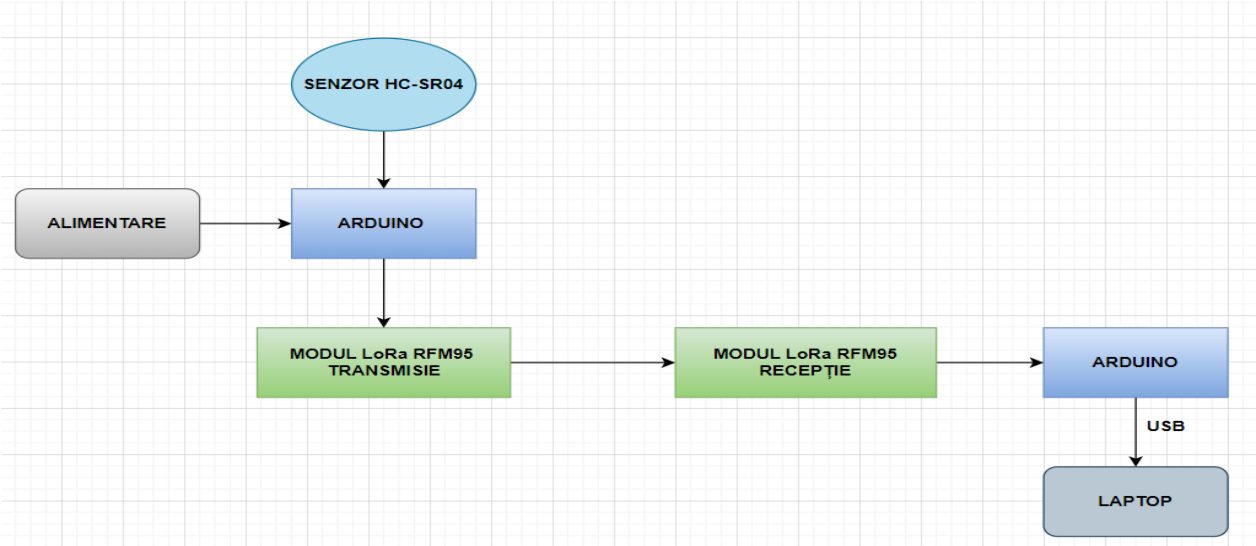

Figura 1 Schema de principiu a sistemului

În Figura 2 este prezentat cablajul efectiv al sistemului, evidențiind modul în care toate componentele hardware sunt conectate între ele pentru a asigura funcționarea corectă a întregului proiect.

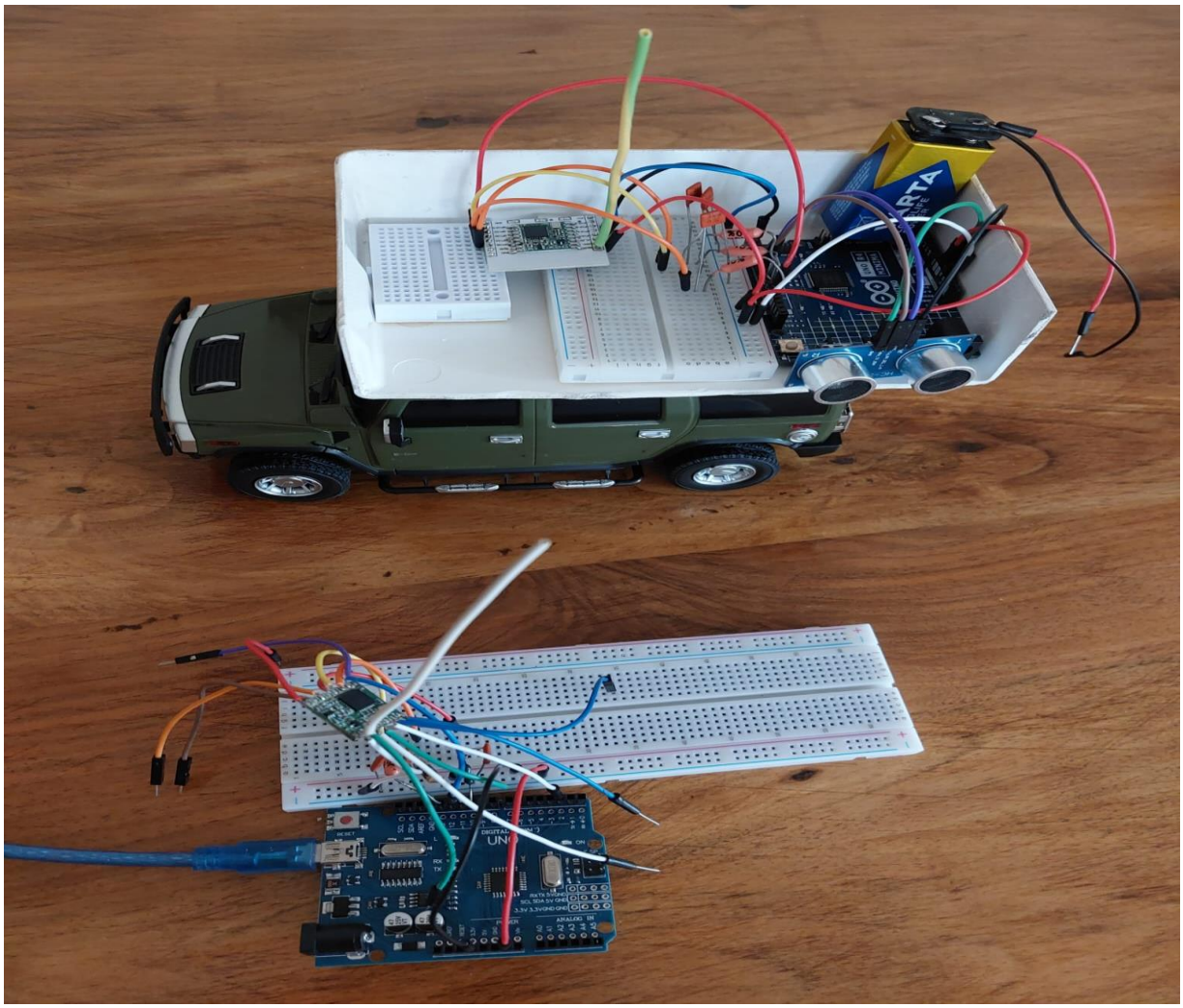

Figura 2 Proiectul la nivel hardware

1.4 MOTIVAȚIA LUCRĂRII

Alegerea unui astfel de proiect reflectă necesitatea de a aborda provocările specifice mediilor subterane și de a dezvolta soluții inovatoare care combină tehnologia hardware și software într-un mod eficient.

Una dintre principalele motivații ale acestui proiect este creșterea siguranței și accesibilității explorării peșterilor. Înainte de a intra într-o peșteră, este crucial să se cunoască detalii precise despre dimensiunile și structura acesteia. Utilizarea senzorilor cu ultrasunete pentru măsurarea distanțelor și obstacolelor permite crearea unei hărți precise a peșterii, ajutând la identificarea pasajelor înguste sau periculoase.

Explorarea peșterilor reprezintă un mediu dificil pentru tehnologia de comunicații și achiziție de date. Utilizarea modulelor LoRa, cunoscute pentru capacitatea lor de a transmite date pe distanțe mari cu consum redus de energie, este ideală pentru acest scop. Prin integrarea acestor module, proiectul demonstrează aplicabilitatea tehnologiei LoRa în scenarii extreme și contribuie la dezvoltarea de soluții robuste și fiabile.

Realizarea unui astfel de proiect oferă oportunități excelente pentru dezvoltarea competențelor tehnice esențiale în domeniul ingineriei. De la proiectarea și integrarea componentelor hardware până la programarea și prelucrarea datelor, proiectul acoperă o gamă largă de abilități practice și teoretice. Aceste competențe sunt extrem de valoroase în contextul actual al industriei tehnologice, unde inovația și eficiența sunt cerințe fundamentale.

2 BAZE TEORETICE

Tehnologia LoRa

Microcontrolere

Senzori

2.1 TEHNOLOGIA LORA

Tehnologia LoRa este o schemă de modulație a spectrului împrăștiat, derivată din modulația Chirp Spread Spectrum (CSS). Aceasta permite schimbul de rată de date pentru sensibilitatea într-o bandă de frecvență fixă, implementând o rată de date variabilă care utilizează factori de răspândire ortogonali. Acest lucru permite designerului de sisteme să optimizeze performanța rețelei într-o lățime de bandă constantă, oferind o combinație ideală între distanță, consum redus de energie și capacitatea de rețea.

LoRa funcționează pe benzi de frezvență ISM, cum ar fi 433 MHz, 868 MHz (Europa) și 915 MHz (America de Nord). Utilizarea acestor benzi neautorizate reduce costurile de implementare și simplifică cerințele de reglementare. LoRa poate transmite date pe distanțe până la 15 km în zone rurale și 5 km în medii urbane dense, fără necesitatea unei linii de vizibilitate directe. [2]

2.1.1 Componentele Sistemului LoRa

1. Modulele LoRa care sunt integrate în dispozitive pentru a permite comunicația LoRa. Ele conțin circuite pentru modulare/demodulare și transceiver.
2. Gateway LoRa care reprezintă un dispozitiv ce recepționează semnalele de la multiple module LoRa și le transmite către un server centralizat sau cloud. Gateway-urile pot gestiona comunicarea bidirecțională și suportă multiple canale de frecvență.
3. Server de rețea LoRa ce reprezintă un server ce gestionează rețeaua LoRaWAN, autentifică dispozitivele și asigură rutarea eficientă a pachetelor de date între dispozitive și aplicații.
4. Aplicații și platforme de vizualizare ce include software-ul utilizat pentru analiza și vizualizarea datelor colectate.

2.1.2 Proprietăți ale Modulației LoRa

Modulația LoRa este scalabilă atât în lățimea de bandă, cât și în frecvență, putând fi utilizată atât pentru aplicații cu bandă îngustă, cât și cu bandă largă. Aceasta poate fi adaptată

ușor pentru ambele moduri de operare prin câteva modificări simple ale registrelor de configurare.

Similar cu modulația FSK, LoRa este o schemă de modulație cu anvelopă constantă, ceea ce înseamnă că aceleași etape PA de înaltă eficiență, cu costuri și consum de energie reduse, pot fi reutilizate fără modificări. Datorită câștigului de procesare asociat cu LoRa, puterea de ieșire a transmițătorului poate fi redusă în comparație cu o legătură FSK convențională, menținând același sau un buget de legături mai bun.

Datorită produsului BT ridicat (BT > 1) și naturii lor asincrone, un semnal LoRa este foarte rezistent la mecanismele de interferență atât in-band, cât și out-of-band. Perioada simbolului LoRa poate fi mai lungă decât durata sistemelor FHSS cu salt rapid, oferind o rezistență excelentă la mecanismele de interferență.

Pulsul chirp este relativ de bandă largă, oferind LoRa rezintență la propagări multi-cale și fading, fiind ideal pentru utilizarea în medii urbane și suburbane.

Pentru o putere de ieșire și o rată de transfer fixe, bugetul de legătură al LoRa depășește cel al FSK convențional. Când este luată în considerare împreună cu rezistența dovedită la interferențe și mecanisme de fading, această îmbunătățire a bugetului de legătură poate duce ușor la o creștere de patru ori mai mare și chiar mai mult în distanță.

O proprietate esențială a LoRa este capacitatea de a distinge clar între erorile de frecvență și timp, fiind ideală pentru aplicațiile radar și, prin urmare, potrivită pentru aplicațiile de localizare și determinarea distanței, cum ar fi serviciile de localizare în timp real.

2.1.3 Principiul Spectrului Împrăștiat

Principiul spectrului împrăștiat este esențial pentru tehnologia LoRa, asigurând comunicații robuste și eficiente. Spectrul împrăștiat se referă la tehnica de răspândire a semnalului pe o bandă de frecvență mai largă decât necesar pentru transmiterea datelor.

Răspândirea semnalului pe o bandă largă de frecvențe face ca aceasta să fie mai puțin susceptibil la interferențe. Interferențele care afectează doar o mică parte din lățimea de bandă nu degradează semnalul general. Aceasta permite detectarea și decodarea semnalelor chiar și în prezența interferențelor semnificative.

Utilizarea spectrului împrăștiat crește semnificativ raportul semnal-zgomot, îmbunătățind astfel fiabilitatea comunicației. Modulația CSS asigură că fiecare bit de informație este distribuit pe o gamă largă de frecvențe, oferind protecție împotriva zgomotului.

Spectrul împrăștiat face ca semnalul să fie mai dificil de detectat și interceptat, oferind un nivel suplimentar de securitate. Tehnologia LoRa utilizează coduri pseudoaleatoare pentru răspândirea semnalului, adăugând un strat de criptare naturală care îmbunătățește securitatea comunicațiilor.

Capacitatea de rezistență la fading și multipath este esențială în mediile urbane și industrial complexe. Semnalele răspândite pot fi recepționate și decodate cu success chiar și în condiții de reflexie și difracție semnificative.

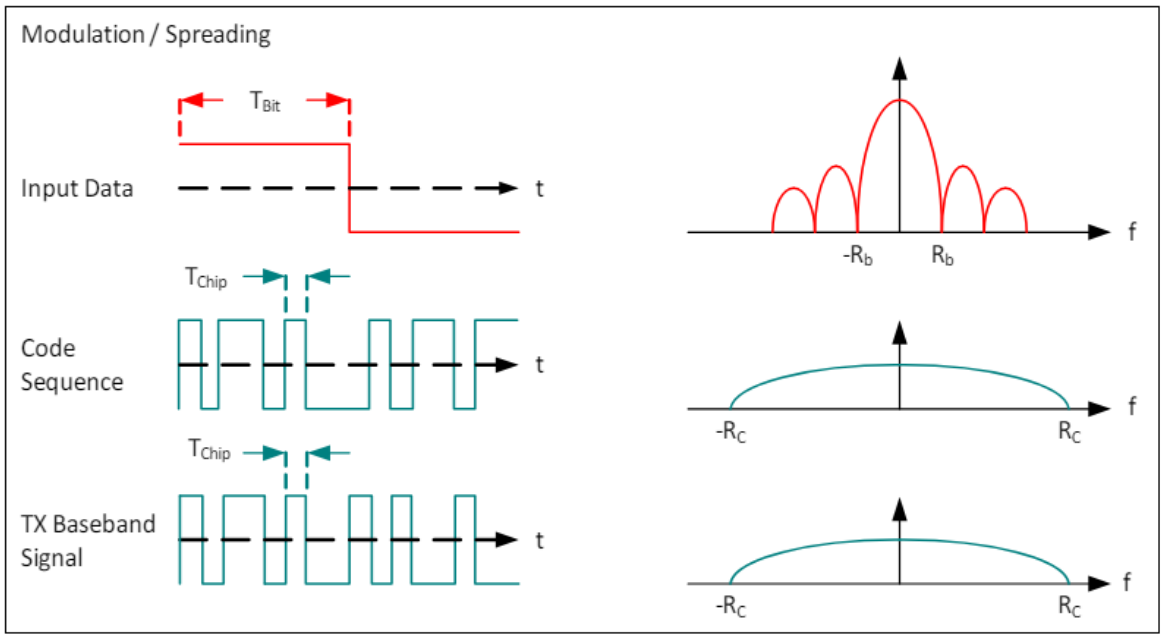

Figura 3 Procesul de modulație/împrăștiere

2.1.4 Modulația și Rata de Date

În modulația LoRa, răspândirea spectrului împrăștiat se realizează prin generarea unui semnal chirp care variază continuu în frecvență. Relația între rata de date dorită, rata simbolurilor și rata de chipuri pentru modulația LoRa poate fi exprimată după cum urmează :

- Rata simbolurilor R_S este reciproca perioadei simbolului T_S:

$$R_s = \frac{1}{Ts} \quad (1)$$

- Rata de chipuri R_C este produsul dintre rata simbolurilor și factorul de răspândire 2^{SF}.
- Rata nominală a datelor R_b este dată de

$$R_b = R_S \cdot (4/4 + CR) \quad (2)$$

unde CR este rata codului.

Modulația LoRa este o implementare simplă de nivel de strat fizic care oferă îmbunătățiri semnificative ale bugetului de legătură în comparație cu modulațiile de bandă îngustă convenționale. [3]

2.1.5 LoRaWAN

Chiar dacă LoRa stabilește baza comunicațiilor la nivel fizic, straturile superioare lipseaua la început. LoRaWAN este un protocol creat pentru a stabili aceste straturi. Acesta este un protocol de control al accesului la mediu (MAC) care se bazează pe cloud, dar acționează în principal ca un protocol de rețea pentru a gestiona comunicarea între gateway-urile LPWAN și dispozitivele finale, fiind utilizat ca un protocol de rutare susținut de către LoRa Alliance.

LoRaWAN controlează frecvențele de comunicație, viteza de date și puterea pentru toate dispozitivele. Dispozitivele din rețea nu transmit date în mod sincronizat, ci doar atunci când este necesar. Informțiile transmise de un dispozitiv final sunt primite de mai multe gateway-uri, care apoi le trimit către un server centralizat de rețea.

În cadrul comunicațiilor wireless, în special în domeniul IoT, este crucială eficiența utilizării canalului și prevenirea coliziunilor pentru asigurarea fiabilității rețelei și eficiența spectrului. În trecut, LoRaWAN a folosit protocolul ALOHA pentru MAC, dar pentru a crește eficiența, LoRa Alliance a adăugat CSMA-CA prin TR013 pentru performanțe îmbunătățite. [2]

2.1.6 Aplicații Practice

Tehnologia LoRa este utilizată într-o varietate de aplicații IoT, inclusiv:

- Monitorizarea mediului
- Agricultură inteligentă
- Gestionarea resurselor
- Smart Cities

2.2 MICROCONTROLERE

2.2.1 Despre microcontrolere

Un microcontroler este un sistem utilizat pentru controlul și procesarea informațiilor provenite de la un proces sau din medii externe. În urma reducerii dimensiunilor, toate componentele necesare controlului sunt integrate pe un singur cip, rezultând un calculator pe un singur cip, specializat în operațiile de control. Microcontrolerul este compus dintr-o unitate centrală de procesare (CPU), memorii (flash, RAM, SRAM, ROM, EEPROM etc.) și periferice

programabile. Acestea sunt utilizate în sisteme integrate încapsulate și în controlul automat al dispozitivelor, precum controlul releelor, sistemelor de alarmă sau a celor de control al motoarelor, în industria electronică și aerospațială.

Inițial, microcontrolerele erau bazate pe logica cablată, având dezavantaje precum dimensiunile mari și consumul energetic crescut. Cu toate acestea, avantajele folosirii lor includ reducerea costurilor de implemenatre și numărul de componente electronice necesare.

Arhitectura unui microcontroler reprezintă proprietățile sale abstracte și implică proiectarea setului de instrucțiuni, organizarea funcțională și implementarea fizică. Aceasta poate fi privită analog cu arhitectura clădirilor, implicând atât aspecte teoretice, cât și practice.

Există mai multe tipuri de arhitecturi de CPU, precum:

- Arhitecturi de tip ''Von Neumann'': Acestea separă spațiile de memorie în două categorii distincte: una pentru program și alta pentru date, cu magistrale de adrese și date separate.
- Arhitecturi CISC (Complex Instruction Set Computer) : Acestea utilizează un set complex de instrucțiuni, permițând folosirea unei singure instrucțiuni mașină pentru fiecare instrucțiune din codul scris într-un limbaj de nivel înalt.
- Arhitecturi RISC (Reduced Instruction Set Computer) : Acestea utilizează un set redus de instrucțiuni, executându-le rapid și eficient. Acest tip de arhitectură permite un circuit mai simplu pentru CPU, ceea ce poate duce la frecvențe de ceas mai mari și este mai adecvat pentru optimizarea codului de către un compilator. [4]

2.2.2 Microcontrolerul folosit – RA4M1

Microcontrolerul Renesas RA4M1 face parte din seria RA și este construit pe nucleul ARM Cortex-M4 cu unitate de punct flotant (FPU). Acesta este destinat aplicațiilor care necesită performanță ridicată și consum redus de energie, incluzând o gamă largă de periferice și funcționalități, fiind ideal pentru aplicații industriale, medicale și de consum.

RA4M1 utilizează nucleul ARM Cortex-M4 cu FPU, având o arhitectură ARMv7E-M și suport pentru instrucțiuni DSP. Acesta funcționează la o frecvență maximă de 48 MHz și include o unitate de protecție a memoriei (MPU) cu 8 regiuni, precum și funcții de debug și trace, cum ar fi ITM, DWT, FPB, TPIU și ETB. Portul de debug CoreSight include JTAG-DP și SW-DP. Memoria internă cuprinde 256 KB pentru cod și 8 KB pentru date, alături de 32 KB de SRAM și cache Flash (FCACHE). Sistemul dispune de un ID unic de 128 biți pentru securitate.

RA4M1 oferă numeroase periferice pentru a facilita diverse aplicații. Acesta include un modul USB 2.0 Full-Speed, cu transceiver integrat și compatibilitate cu specificația de încărcare

USB 1.2. Serial Communications Interface (SCI) are 4 canale pentru UART, I^2C simplu și SPI simplu, iar SPI are 2 canale. De asemnea, există 2 canale pentru I^2C și un modul CAN pentru comunicații electromagnetice zgomotoase. Interfața serială pentru sunet suportă PCM audio.

Microcontrolerul include un convertor A/D de 14 biți, care poate procesa până la 25 de canale de intrare analogică, iar acuratețea conversiei este selectabilă între 12 și 14 biți. Convertorul D/A de 12 biți are un amplificator de ieșire, iar convertorul D/A de 8 biți este utilizat pentru referințele comparatorului analogic de joasă putere. Senzorul de temperatură integrat monitorizează temperatura cipului, iar amplificatorul operațional are patru unități pentru amplificarea tensiunilor analogice mici.

Microcontrolerul dispune de două module de General PWM Timer, unul de 32 de biți și altul de 16 biți, care sunt utilizate pentru generarea de unde PWM și pentru controlul motoarelor DC fără perii. De asemenea, include două module de timer asincron de joasă putere și un watchdog timerpentru monitorizarea sistemului și prevenirea funcționării defectuoase.

RA4M1 integrează mai multe funcții de securitate și siguranță, inclusiv corecția erorilor codului pentru SRAM, protecția memoriei Flash, funcții de diagnosticare a ADC-ului și măsurarea acurateței frecvenței ceasului. Modulul de securitate include AES128/256, un generator de numere aleatoare reale și GHASH pentru criptare și securitate sporită.

Pentru gestionarea consumului de energie, microcontrolerul oferă moduri de consum redus, precum și un ceas în timp real cu suport pentru calendar și backup pe baterie. Controller-ul de Evenimente Link permite interacțiunea directă între module fără intervenția CPU, iar controller-ul DMA și controller-ul de transfer de date facilitează transferul eficient al datelor.

Microcontrolerul RA4M1 include un controler pentru afișaje LCD segment, care suportă moduri de afișsare multiple, și o unitate de senzație tactilă capacitivă pentru măsurarea capacității electrostatice și detectarea atingerilor.

RA4M1 dispune de mai multe surse de ceas, inclusiv un oscilator principal (MOSC) cu frecvențe între 1 și 20 MHz, un oscilator secundar (SOSC) de 32.768 KHz și oscilatoare integrate pe cip (HOCO, MOCO, LOCO) cu diverse frecvențe. De asemenea, există un oscilator dedicat pentru IWDT de 15KHz și funcții de ajustare a ceasului.

În Figura 4 este reprezentată diagrama bloc a subsetului de microcontrolere. Diagrama arată o reprezentare a funcționalităților generale disponibile. [5]

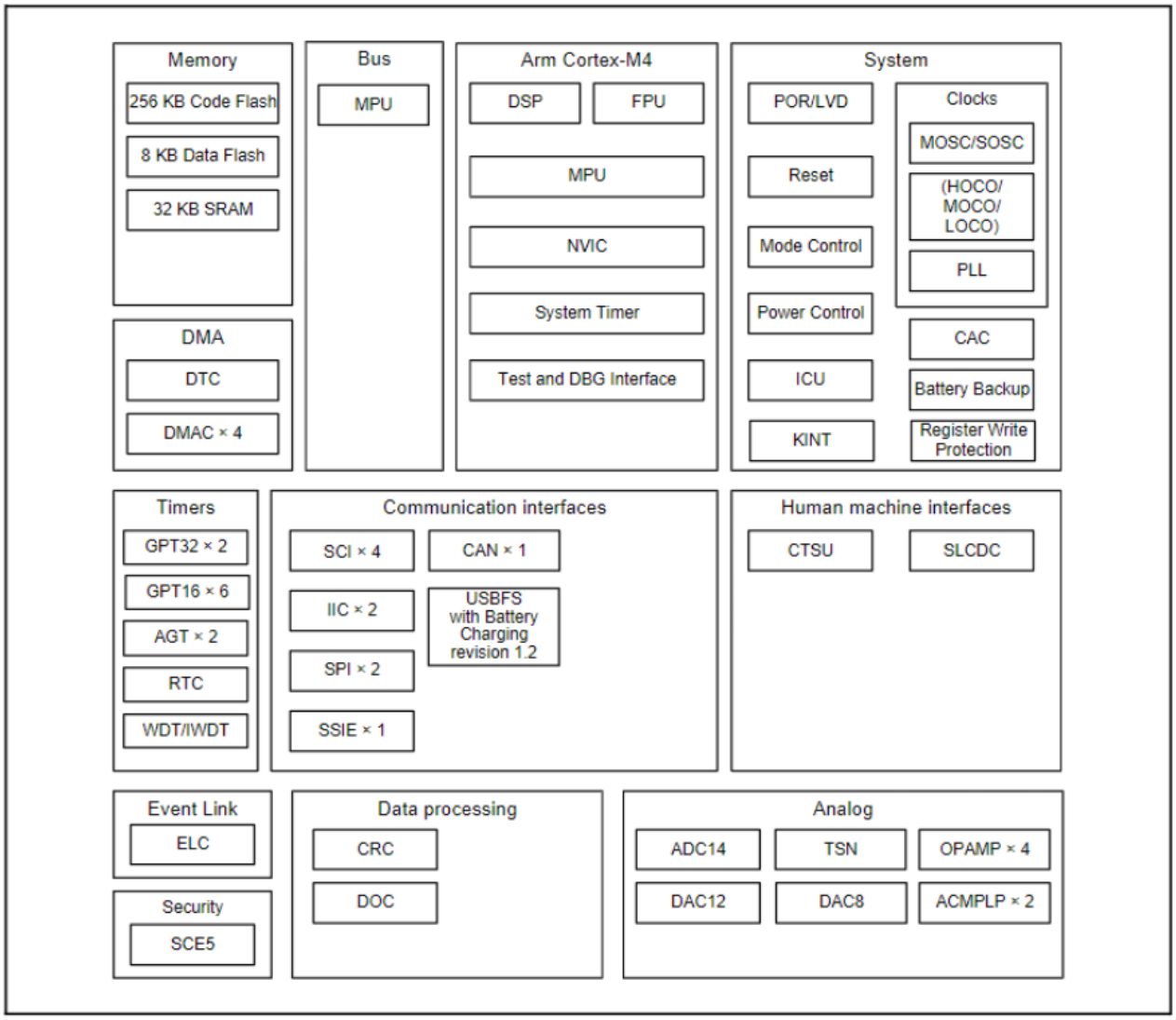

Figura 4 Diagrama bloc a RA4M1 [5]

Microcontrolerul oferă până la 84 de pini I/O, inclusiv pini toleranți la 5V și pini de intrare/ieșire CMOS, care pot fi folosiți pentru conectarea la diverse dispozitive periferice și pentru generarea de semnale de control.

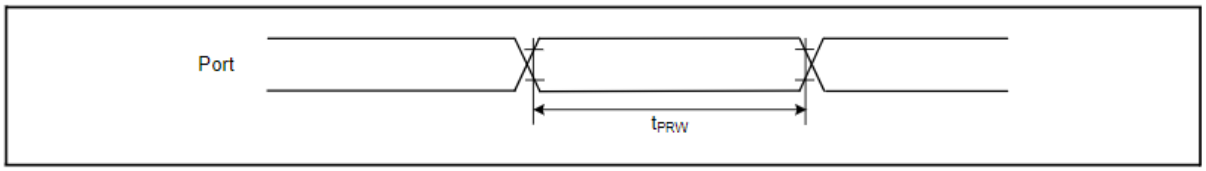

Figura 5 Timpul de intrare pentru porturile I/O [5]

În Figura 6 se poate observa alocarea pinilor pentru QFP de 64 de pini. Am ales să listez figura acestui microcontoler cu codul de identificare RA7FA4M1AB3CFM deoarece acesta este cipul pe care îl are plăcuța mea ARDUINO UNO.

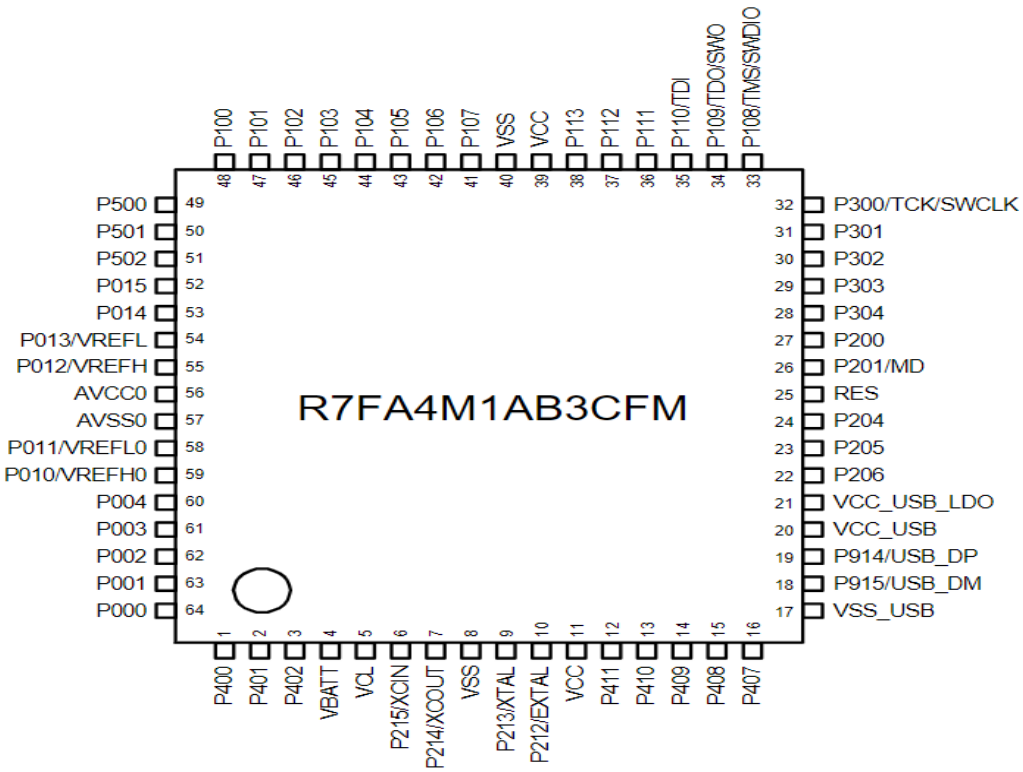

Figura 6 Alocarea pinilor pentru QFP de 64 de pini [5]

2.2.3 Microcontrolerul folosit – ATmega328P

ATmega328P este un microcontroler CMOS pe 8 biți, de putere redusă, bazat pe arhitectura AVR RISC. Acesta este capabil să execute instrucțiuni puternice într-un singur ciclu de ceas, permițând atingerea unor rate de transfer de până la 1MIPS per MHz. Aceasta optimizează echilibrul dintre consumul de energie și viteza de procesare, aspect crucial în proiectele embedded.

Arhitectura RISC avansată a ATmega328P include 131 de instrucțiuni puternice, majoritatea fiind executate într-un singur ciclu de ceas. Având un set extins de instrucțiuni, microcontrolerul poate gestiona sarcini complexe într-un mod eficient. În plus, microcontrolerul dispune de 32K bytes de memorie flash programabilă în sistem, 1K byte de EEPROM și 2K bytes de SRAM internă. Aceste capacități de memorie non-volatilă de mare capacitate permit stocarea și gestionarea eficientă a codului și datelor critice.

ATmega328P integrează și două temporizatoare/contorizatoare pe 8 biți și unul pe 16 biți, fiecare având un prescalator separat și moduri de comparare. De asemenea, include șase canale PWM și opt canale ADC pe 10 biți, care permit conversia analog-digitală precisă și generarea semnalelor PWM pentru diverse aplicații. În ceea ce privește interfețele seriale,

microcontrolerul dispune de USART programabil, interfață serială master/slave SPI și interfață serială pe 2 fire (compatibilă I^2C).

ATmega328P este disponibil în mai multe configurații de pachete, inclusiv TQFP și QFN/MLF, iar configurarea pinilor este esențială pentru înțelegerea modului de conectare și utilizare a microcontrolerului. Configurația pinilor pentru pachetul TQFP și pentru pachetul QFN/MLF arată poziția și funcționalitatea fiecărui pin, facilitând designul hardware. Aceste configurări sunt prezentate în Figura 7.

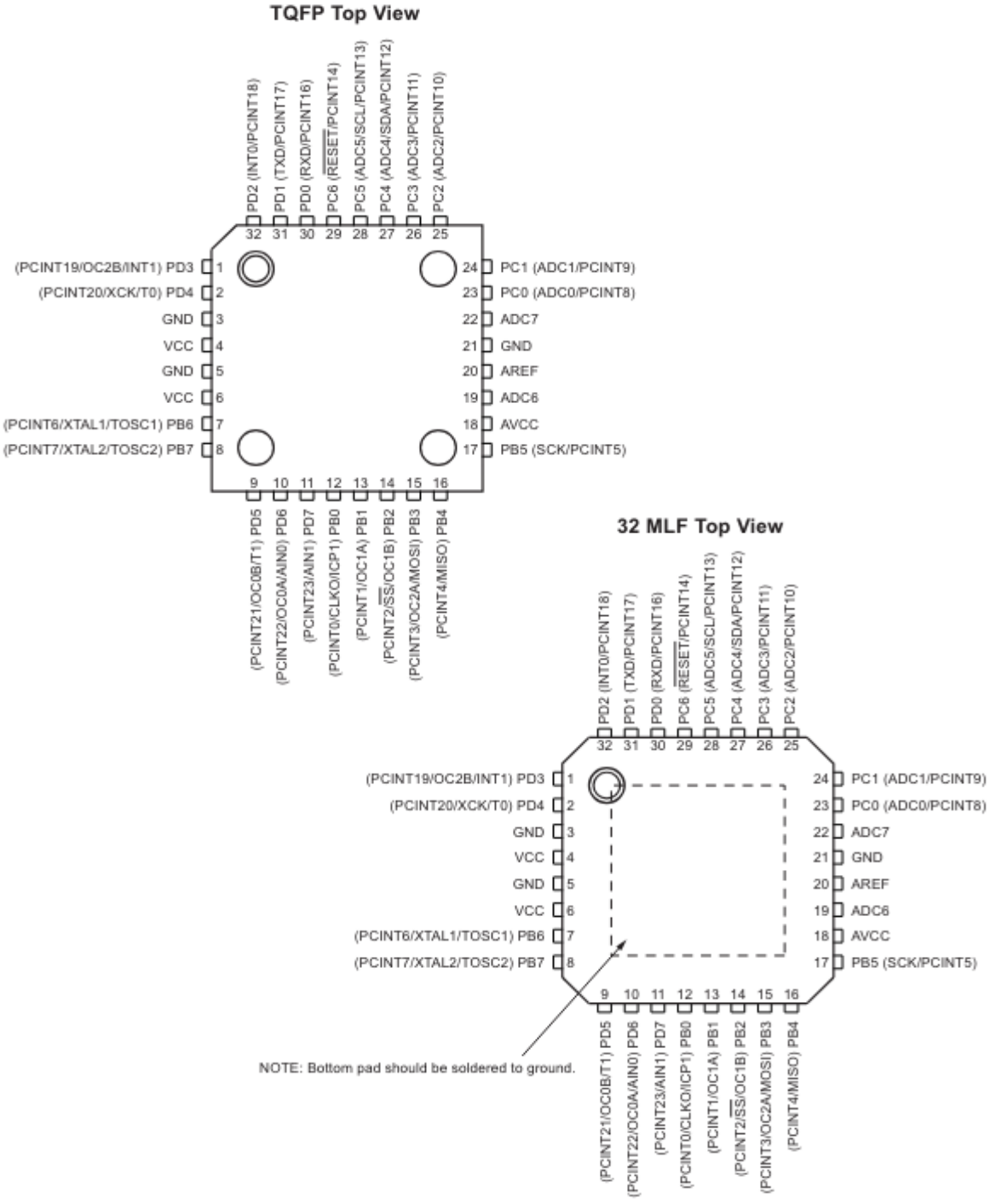

Figura 7 Configurarea pinilor pentru pachetul TQPF și pachetul MLF [6]

Funcțiile speciale ale microcontrolerului includ surse de alimentare și resetare, cum ar fi resetarea la pornire și detecția programabilă de brown-out, asigurând o pornire corectă și protecție împotriva fluctuațiilor de tensiune. Microcontrolerul dispune de șase moduri de somn: Idle, reducerea zgomotului ADC, economisire de energie, oprire, standby și standby extins, permițând optimizarea consumului de energie în funcție de necesitățile aplicației. Programarea în sistem este facilitată prin intermediul unui program de boot pe chip, suportând operațiuni de citire-în-timp-ce-scrie, ceea ce facilitează actualizările de firmware.

Diagrama bloc a ATmega328P, prezentată în Figura 8, oferă o perspectivă clară asupra arhitecturii interne a microcontrolerului, incluzând nucleul AVR, memoriile interne, modulele periferice și circuitele de ceas și resetare. Aceasta evidențiază componentele și interconectările lor, oferind o înțelegere detaliată a structurii și funcționării interne a microcontrolerului. [6]

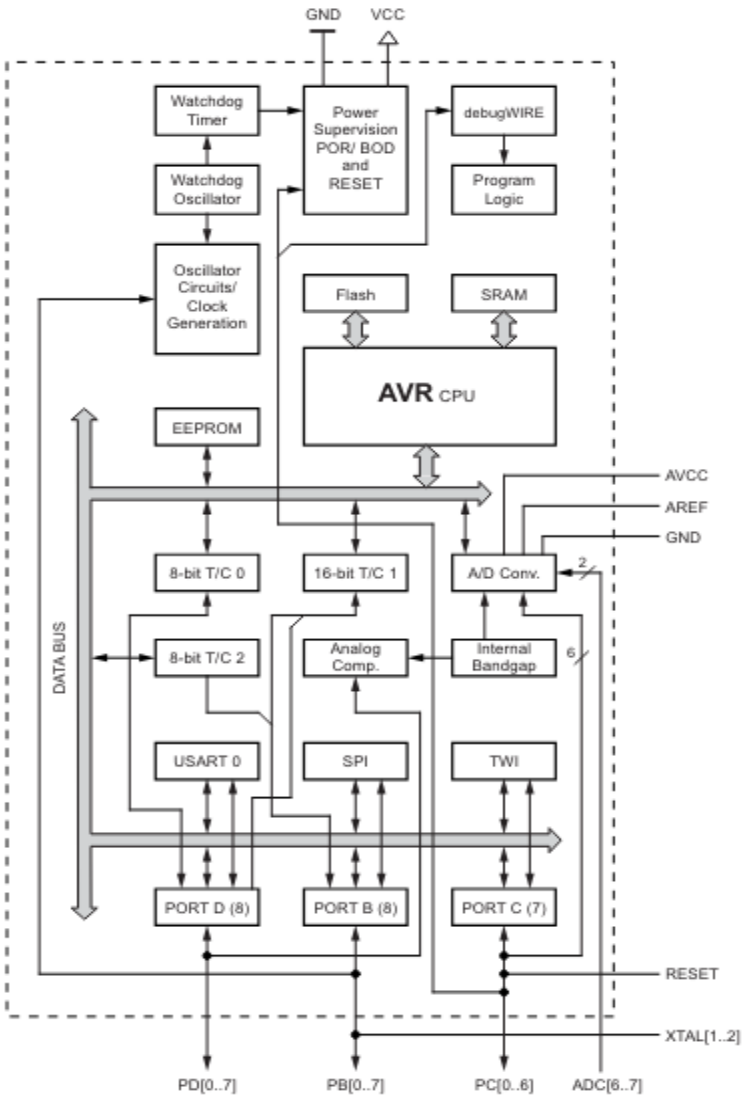

Figura 8 Diagrama bloc a microcontrolerului ATmega328P [6]

ATmega328P dispune de memorie flash de 32K bytes reprogramabilă în sistem, utilizată pentru stocarea codului și a datelor critice, 1K byte de EEPROM pentru stocarea datelor nevolatile care trebuie păstrate între ciclurile de alimentare și 2K bytes de SRAM pentru stocarea temporară a datelor în timpul execuției. Nucleul AVR combină un set de instrucțiuni bogat cu 32 de registre de lucru cu scop general. Toate registrele sunt conectate direct la unitatea aritmetică și logică (ALU), permițând accesul independent la două registre într-o singură instrucțiune, ceea ce duce la o eficiență sporită a codului și la o viteză de execuție semnificativă.

ATmega328P este ideal pentru aplicații variate, de la automatizări industriale la proiecte DIY și sisteme embedded. Flexibilitatea și performanța sa îl fac potrivit pentru gestionarea senzorilor și modulelor de comunicație în proiecte complexe.

2.3 SENZORI

2.3.1 Ce este un senzor?

Senzorii constituie componente esențiale în rețelele fără fir, fiind de asemnea, utilizați în proiectul implementat. Un senzor poate fi definit ca un dispozitiv care măsoară diverse caracteristici fizice sau chimice și convertește aceste măsurători în semnale care pot fi interpretate de un observator sau de un sistem automat de achiziție a datelor. Aceste dispozitive sunt familiar și folosite în mod cotidian: un exemplu simplu de senzor este termometrul, folosit pentru a măsura temperatura.

Pentru a obține măsurători de la un senzor, există două modalități: utilizarea unei scale de referință integră în senzor (așa cum se întâmplă în cazul unui termometru) sau citirea unui semnal digital/analogic generat de senzor. În ultimul caz, senzorul este adesea asociat cu un dispozitiv de măsurare, care interpretează semnalul măsurat. Deși termenii „senzori” și „metri” sunt adesea folosiți interschimbabil, aceștia reprezintă concepte diferite: senzorii detectează o anumită condiție fizică sau chimică și generează un semnal, în timp ce metrii sunt folosiți pentru a măsura și afișa acest semnal într-un format ușor de înțeles. Adesea, senzorii și metrii sunt combinați într-un singur instrument, cum este cazul senzorului VP3, care măsoară temperatura și umiditatea relativă și calculează deficitul presiunii vaporilor.

Senzorii analogici generează, de obicei, un semnal de ieșire sub formă de tensiune sau curent în răspuns la caracteristici precum lumina sau temperatura. Acest semnal poate fi măsurat folosind metrii adecvați. În schimb, senzorii digitali includ adesea un microprocesor care prelucrează semnalul și efectuează calcule. Acești senzori sunt cunoscuți sub numele de

„senzori inteligenți” și sunt din ce în ce mai populari, odată cu scăderea costurilor și îmbunătățirea performanțelor microprocesoarelor. [7]

Înregistrarea datelor este esențială atunci când se utilizează senzori pentru măsurători. Deși este posibil să se înregistreze manual toate datele, acest lucru poate fi ineficient și prezintă riscul de erori. În cazul în care datele obținute de la un senzor sunt importante, este util să existe un dispozitiv care să poată stoca și exporta datele într-un format ușor de utilizat, cum ar fi o foaie de calcul sau un program de baze de date. Metrii portabili sunt, de obicei, disponibili pentru măsurarea și stocarea datelor, fiind conectați la un calculator folosind un cablu USB.

2.3.2 Acuratețe/Precizie/Rezoluție

Colectarea datelor de orice fel nu este utilă dacă nu știm cât de precise sunt aceste date. În multe cazuri, este mai bine să nu avem date, decât să avem date inexacte sau eronate, deoarece acestea pot conduce la decizii greșite. Prin urmare, înțelegerea acurateții senzorilor este importantă, dar din păcate, nu întotdeauna este ușor de determinat. Există mai multe concepte de bază pe care utilizatorii trebuie să le cunoască pentru a evolua acuratețea unui senzor. Producătorii de renume, de obicei, includ aceste informații în specuficațiile senzorilor.

Precizia unei măsurători reprezintă gradul în care măsurătorile repetate, în condiții neschimbate, furnizează aceleași rezultate. Cu alte cuvinte, dacă măsurăm aceeași entitate de zece ori, cât de apropiate sunt cele zece rezultate? Nu toate măsurătorile vor fi identice, dar cât de diferite sunt ele? Este important de menționat că precizia este diferită de acuratețe: este posibil ca zece măsurători să fie foarte similare, dar toate să fie greșite. În acest caz, măsurătorile sunt precise, dar nu sunt exacte. Precizia poate fi descrisă de două componente separate:

- Repetabilitate: variația măsurătorilor în condiții constante, în timp ce măsurătorile sunt repetate într-un interval scurt de timp.
- Reproducibilitate: variația măsurătorilor în timpul utilizării aceluiași proces de măsurare pe o perioadă îndelungată, în rândul mai multor senzori/metri și operatori.

În Figura 9 sunt prezente cele patru posibile scenarii acuratețe și precizie. Punctele cu negru reprezintă măsurătorile, iar puncul cu galben din centru reprezintă valoarea reală.

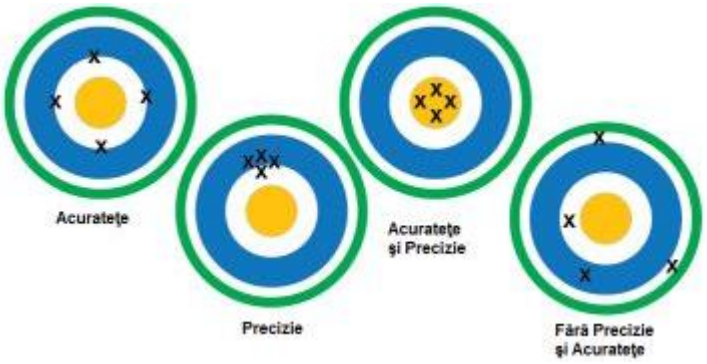

Figura 9 Cele 4 scenarii de acurateţe și precizie [8]

Ideal, un senzor ar trebui să aibă atât acurateţe, cât și precizie ridicată. Acurateţea poate fi îmbunătățită prin calibrare, în timp ce precizia nu poate fi ajustată.

Rezoluția unui senzor se referă la cea mai mică schimbare detectabilă în cantitatea măsurată. De exemplu, cântarele de bucătărie ieftine pot detecta modificări foarte mici în greutate. Dacă cea mai mică schimbare detectabilă este de 0,01 g, atunci aceasta este rezoluția cântarului. Majoritatea senzorilor moderni au o rezoluție mai bună decât precizia și acurateţea lor. Un cântar de bucătărie poate afișa greutăți cu o rezoluție de 0,01 g, în timp ce precizia și acurateţea sa pot fi de 1g.

Pentru rezultate de calitate, specificațiile senzorilor trebuie să includă nu doar o rezoluție ridicată, ci și o precizie și acurateţe adecvată.

3 COMPONENTE HARDWARE UTILIZATE

Listă echipamente utilizate

Prezentare echipamente utilizate

3.1 LISTĂ ECHIPAMENTE UTILIZATE

Sistemul prezintă o serie de echipamente hardware care au fost folosite în realizarea proiectului. Acestea sunt prezente în următoarea listă:

- Arduino UNO R4 Minima;
- Arduino UNO R3;
- Modul LoRa RFM95;
- Senzor cu ultrasunete – HC-SR04;
- Rezistoare – Divizor de tensiune;
- Placă adaptoare pentru module – ESP8266;
- Sursă alimentare – baterie de 9V;
- Cablu USB;
- Breadboard;
- Conectori mama-tată;
- Soclu baterie

3.2 PREZENTARE ECHIPAMENTE UTILIZATE

3.2.1 Plăcuța Arduino UNO R4 Minima

Arduino UNO este una dintre cele mai populare plăci de dezvoltare din gama Arduino, cunoscută pentru simplitatea și versatilitatea sa. De la lansarea primei versiuni, aceasta a evoluat semnificativ, adaptându-se la cerințele tot mai complexe ale utilizatorilor și proiectelor 26rimat. Seria R4 reprezintă un pas major în această evoluție.

Arduino UNO R4 Minima este prima placă din seria UNO care dispune de un microcontroller pe 32 de biți. Acesta utilizează un microcontroller din seria RA4M1 de la Renesas (R7FA4M1AB3CFM), care integrează un nucleu Arm Cortex-M4 de 48 MHz. Cpmparativ cu predecesorii săi, UNO R4 Minima oferă o memorie semnificativ mai mare: 256 kB memorie flash, 32kB SRAM și 8kB memorie EEPROM. Aceste îmbunătățiri aduc beneficii

considerabile în ceea ce privește capacitățile de procesare și stocare, făcând placa potrivită pentru aplicații mai complexe.

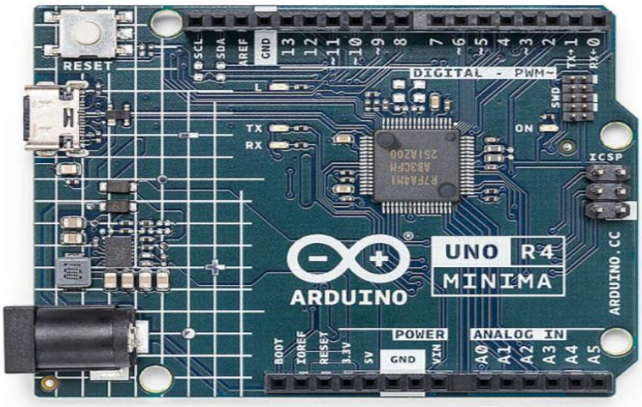

Figura 10 Placa de dezvoltare Arduino UNO R4 Minima

A. Arhitectura și specificații
    1. Microprocesor
        - Nucleu CPU: ARM Cortex-M4 cu unitate de punct flotant
        - Frecvență de lucru: 48MHz
        - Memorie: 256 kB pentru memoria flash care este utilizată pentru stocarea programului. Pentru memoria SRAM 32 kB utilizată pentru stocarea temporară a datelor în timpul execuției. EEPROM este de 8 kB care este utilizată pentru stocarea permanentă a datelor de configurare.
    2. Periferice și funcționalități
        - Real-Time Clock: Integrat, permite gestionarea precisă a timpului
        - Memory Protection Unit: Protejează accesul la memorie, asigurând securitatea și stabilirea aplicațiilor
        - Digital Analog Converter: Inclus, cu o rezoluție de până de 12 biți, permite conversia semnalelor digitale în semnale analogice

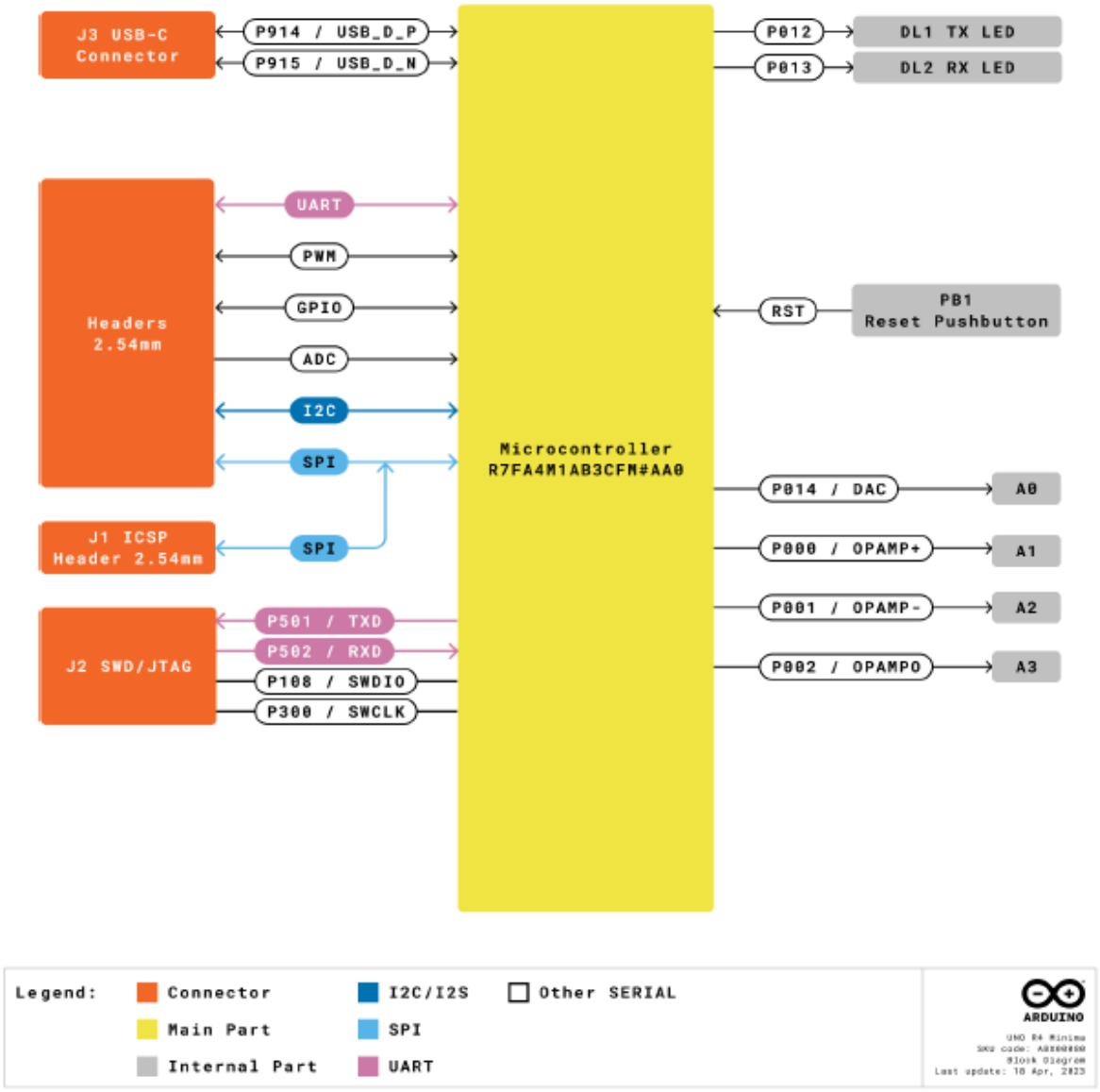

Figura 11 Diagrama bloc a plăcii Arduino UNO R4 Minima [9]

B. Pini I/O și conectori

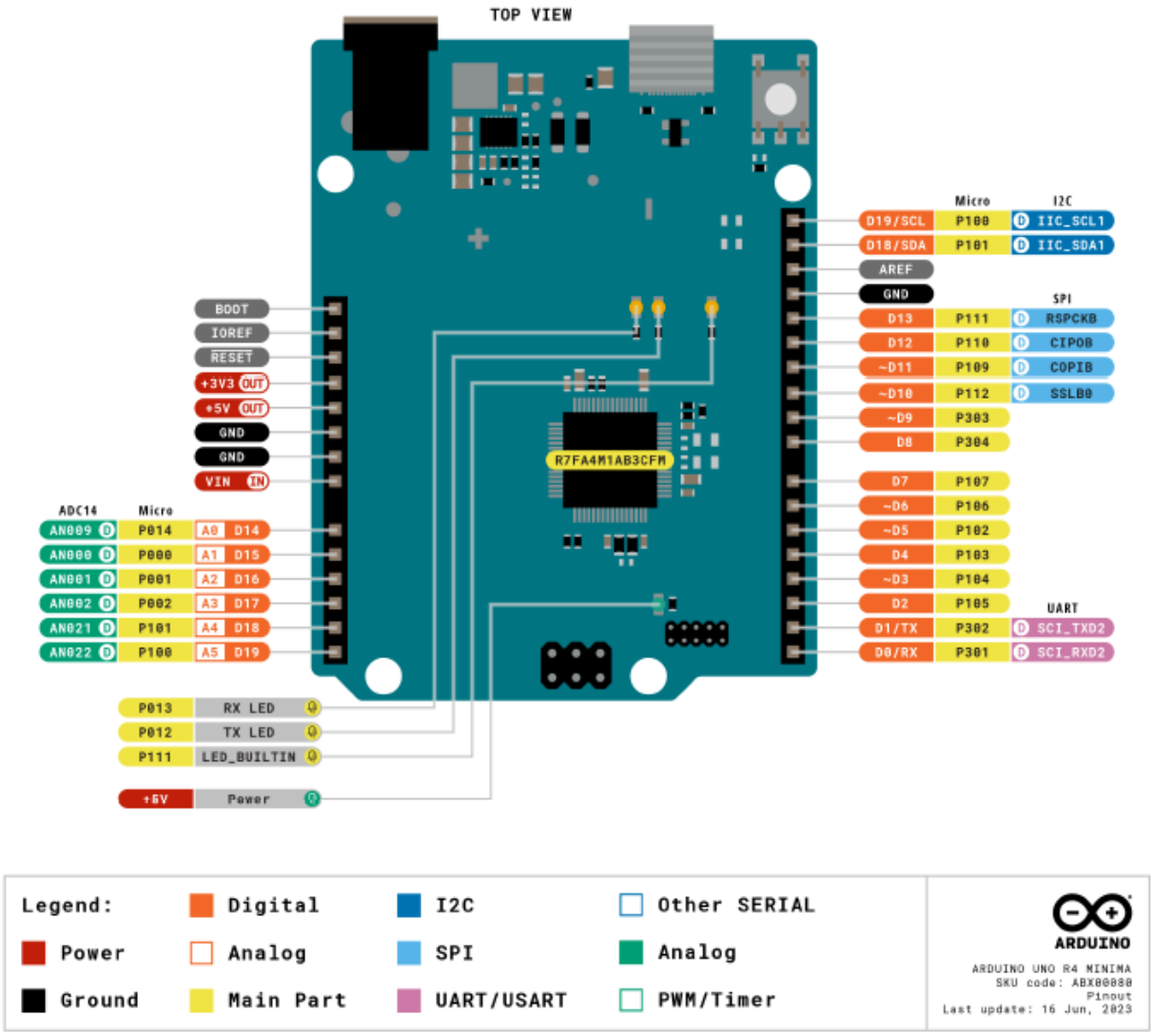

Figura 12 Configurația pinilor a plăcii Arduino UNO R4 Minima [9]

1. Pini digitali și analogici
   - 14 pini digitali (GPIO): Numerotați D0-D13, utilizabili pentru intrare și ieșire

Tabelul 1 Pinii digitali a Arduino UNO R4 Minima

Pin	Funcție	Tip	Descriere
1	SCL	Digital	Ceas Serial I²C
2	SDA	Digital	Date Seriale I²C
3	AREF	Digital	Tensiune de referință analogică
4	GND	Power	Masă
5	D13/SCK	Digital	Ceas SPI/GPIO 13

6	D12/CIPO	Digital	GPIO12/ Controler SPI In Periferic Out
7	D11/COPI	Digital	GPIO11 (PWM)/Controler SPI Out Periferic In
8	D10/CS	Digital	GPIO10(PWM)/Selectare cip SPI
9	D9	Digital	GPIO9(PWM)
10	D8	Digital	GPIO 8
11	D7	Digital	GPIO 7
12	D6	Digital	GPIO 6 (PWM)
13	D5/CANRX0	Digital	GPIO5(PWM)/Transmițător CAN (TX)
14	D4/CANTX0	Digital	GPIO 4/Receptor CAN (RX)
15	D3	Digital	GPIO 3 (PWM)/Pin de întrerupere
16	D2	Digital	GPIO 2/Pin de întrerupere
17	D1/TX0	Digital	GPIO 1/Transmițător Serial 0 (TX)
18	D0/TX0	Digital	GPIO 0/Receptor Serial 0 (RX)

- 6 canale de intrare analogică: Numerotate A0-A5, permit măsurarea semnalelor analogice.

Tabelul 2 Pinii analogici ai Arduino UNO R4 Minima

Pin	Funcție	Tip	Descriere
1	BOOT	MD	Selectarea modului
2	IOREF	IOREF	Referință pentru logica digital V-conectat la 5V
3	Reset	Reset	Resetare
4	+3V3	Power	Rail de alimentare +3V3
5	+5V	Power	Rail de alimentare +5V
6	GND	Power	Masă
7	GND	Power	Masă

8	VIN	Power	Tensiune de intrare
9	A0	Analog	Intrare analogică 0 / DAC
10	A1	Analog	Intrare analogică 1 / OPAMP+
11	A2	Analog	Intrare analogică 2 / OPAMP-
12	A3	Analog	Intrare analogică 3 / OPAMPOut
13	A4	Analog	Intrare analogică 4 / Date seriale I^2C (SDA)
14	A5	Analog	Intrare analogică 5 / Ceas serial I^2C (SCL)

2. Pini PWM
   - 6 pini PWM: D3, D5, D6, D9, D10, D11, utilizabili pentru generarea semnalelor cu modulare în lățime de impuls.
3. Interfețe de comunicare
   - 1 x UART: Disponibil pe pinii D0 și D1, utilizat pentru comunicație serială.
   - 1 x SPI: Disponibil pe pinii D10-D13 și pe header-ul ICSP, utilizat pentru comunicație serială de mare viteză.
   - 1 x I^2C: Disponibil pe pinii A4 și A5, utilizat pentru comunicație între microcontrolere și alte dispozitive I^2C.
   - 1 x CAN: Disponibil pe pinii D4 și D5 (necesită transceiver extern), utilizat în aplicații auto și industrial pentru comunicație robustă și eficientă

C. Periferice integrate
1. Capacitive touch sensing unit (CTSU): Permite detectarea atingerii capacitive, utilă în interfețele tactile.
2. USB 2.0 full – speed module (USBFS): Permite comunicația rapidă și eficientă prin portul USB.
3. Convertor Analog-Digital (ADC): Convertorul ADC integrat oferă o rezoluție de până la 14 biți, asigurând măsurători precise ale semnalelor analogice.
4. Convertor Digital-Analog (DAC): Convertorul DAC integrat are o rezoluție de până la 12 biți, permițând generarea precisă a semnalelor analogice din date digitale.
5. Amplificator operațional (OPAMP): Include un amplificator operațional integrat, util în aplicații de amplificare și filtrare a semnalelor.

D. Opțiuni de alimentare
1. Metode de alimentare
   - VIN: Tensiune de intrare între 6-24V.

- USB-C: Tensiune de operare de 5V, utilizat pentru alimentare și programare.
- Jack Baril: Conectat la pinul VIN pentru alimentare externă.

2. Protecții de alimentare

   Dioda Schottky protejează placa împotriva supratensiunii și polarității inverse.

E. Consumul de curent

Curentul de operare este tipic 33.39 mA când este alimentată prin USB-C și rulează firmware-ul implicit.

F. Funcționare și conectivitate

Conectorul USB-C este utilizat pentru alimentare, programare și comunicație serială, oferind o conectivitate rapidă și ușoară.

Placa include LED-uri pentru indicarea activității de transmisie, recepție, alimentare și pentru semnalizarea stării SCK.

G. Programarea

Programarea plăcii Arduino UNO R4 Minima este facilitată de mediul Arduino IDE, care oferă un cadru intuitiv și accesibil pentru dezvoltarea aplicațiilor.

Arduino UNO R4 Minima combină simplitatea și accesibiltatea caracteristicilor plăcilor Arduino cu puterea și flexibilitatea unui microcontroler modern pe 32 de biți. Aceasta reprezintă o soluție ideală pentru o gamă largă de aplicații, de la proiecte educaționale și hobby până la proiectări industriale. [9]

3.2.2 Plăcuța Arduino UNO R3

Arduino UNO R3 reprezintă una dintre cele mai populare plăci de dezvoltare utilizate în proiectele de electronică și programare. Această placă versatilă este echipată cu procesorul Atmega328P și coprocesorul ATMega16U2, oferind o platformă robustă pentru inițierea în lumea Arduino și pentru dezvoltarea de proiecte complexe.

Figura 13 Placa de dezvoltare Arduino UNO R3

Caracteristicile principale ale plăcii includ un microcontroler AVR cu arhitectură RISC pe 8 biți, capabil să funcționeze la o frecvență de până la 20 MHz. Atmega328P dispune de 32KB de

memorie flash, 2KB de SRAM și 1KB de EEPROM, asigurând suficientă capacitate de stocare pentru majoritatea aplicațiilor. Placa include, de asemenea, 14 pini digitali de intrare/ieșire, dintre care 6 pot fi utilizați ca ieșiri PWM, și 6 intrări analogice. Configurația pinilor Arduino UNO R3 este prezentată în Figura 14.

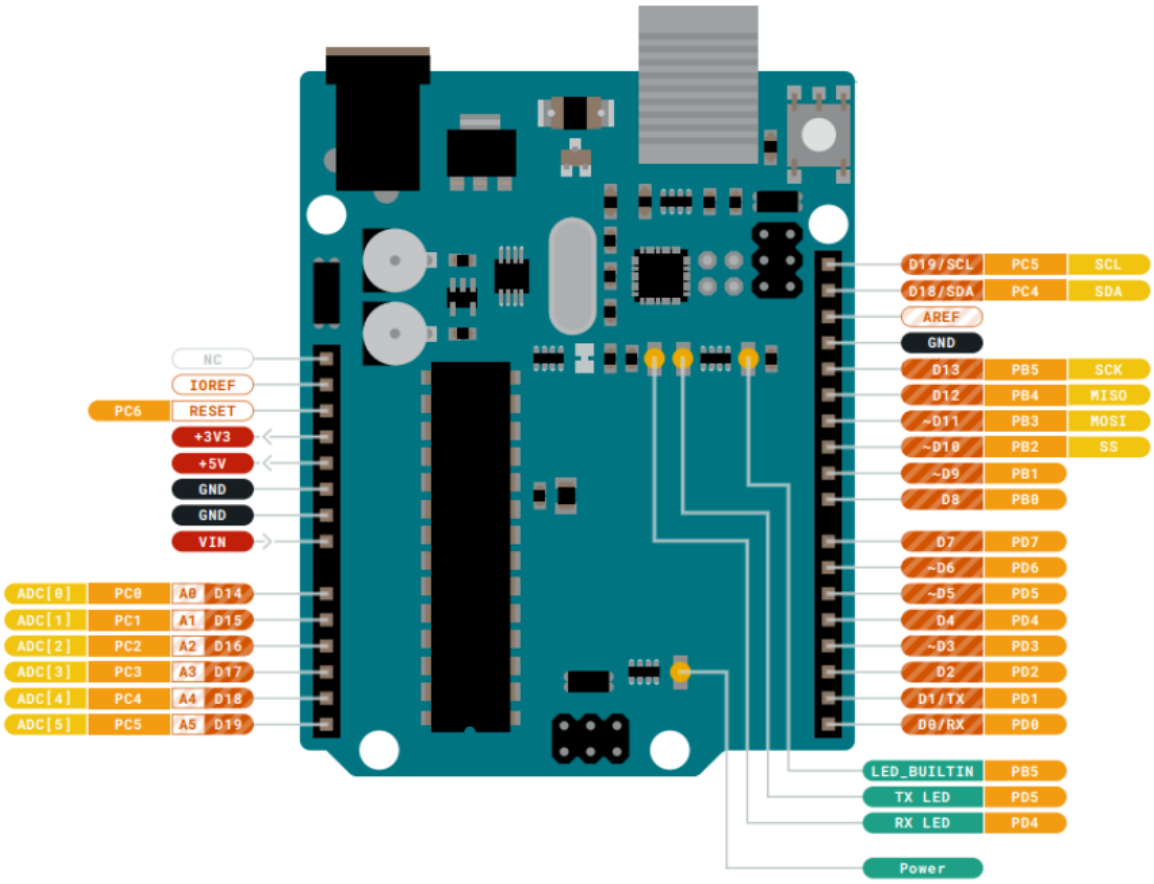

Figura 14 Configurația pinilor Arduino UNO R3 [10]

Arduino UNO R3 dispune de un set variat de pini care permit interconectarea cu diverse module și senzori. Configurația pinilor analogici și digitali este esențială pentru funcționarea corectă a plăcii în cadrul proiectelor. Pinii analogici sunt utilizați pentru citirea semnalelor de la senzori analogici și pentru comunicații I^2C. Configurația pinilor analogici este detaliată în Tabelul 3.

Tabelul 3 Configurația pinilor analogici Arduino UNO R3

Pin	Funcție	Tip	Descriere
1	NC	NC	Neconectat
2	IOREF	IOREF	Referință pentru logica digitală V-conectată la 5V
3	Reset	Reset	Reset

4	+3V3	Power	+3V3 Sursa de alimentare
5	+5V	Power	+5V Sursa de alimentare
6	GND	Power	Masă
7	GND	Power	Masă
8	VIN	Power	Intrare de tensiune
9	A0	Analog/GPIO	Intrare analogică 0 / GPIO
10	A1	Analog/GPIO	Intrare analogică 1 / GPIO
11	A2	Analog/GPIO	Intrare analogică 2 / GPIO
12	A3	Analog/GPIO	Intrare analogică 3 / GPIO
13	A4/SDA	Analog input/I^2C	Intrare analogică 4 / Linie de date I^2C
14	A5/SCL	Analog input/I^2C	Intrare analogică 5 / Linie de date I^2C

Pinii digitali, pe de altă parte, pot fi utilizați pentru intrări/ieșiri digitale și pentru funcții de comunicație serială, SPI și I^2C. Configurația detaliată a pinilor digitali este prezentată în Tabelul 4.

Tabelul 4 Configurația pinilor digitali Arduino UNO R3

Pin	Tip	Funcție	Descriere
1	D0	Digital/GPIO	Pin digital 0 / GPIO
2	D1	Digital/GPIO	Pin digital 1 / GPIO
3	D2	Digital/GPIO	Pin digital 2 / GPIO
4	D3	Digital/GPIO	Pin digital 3 / GPIO
5	D4	Digital/GPIO	Pin digital 4 / GPIO
6	D5	Digital/GPIO	Pin digital 5 / GPIO
7	D6	Digital/GPIO	Pin digital 6 / GPIO
8	D7	Digital/GPIO	Pin digital 7 / GPIO
9	D8	Digital/GPIO	Pin digital 8 / GPIO
10	D9	Digital/GPIO	Pin digital 9 / GPIO
11	SS	Digital	Selectare cip SPI
12	MOSI	Digital	SPI principal la secundar IN
13	MISO	Digital	SPI principal la secundar OUT
14	SCK	Digital	Ieșire de ceas serial SPI
15	GND	Power	Masă

16	AREF	Digital	Tensiune de referință analogică
17	A4/SD4	Digital	Intrare analogică 4 / Linie de date I²C (duplicat)
18	A5/SD5	Digital	Intrare analogică 5 / Linie de date I²C (duplicat)

Un element esențial al plăcii Arduino UNO R3 este varietatea de periferice și interfețe pe care le oferă. Placa dispune de două temporizatoare/contorizatoare pe 8 biți și unul pe 16 biți, fiecare cu registre dedicate pentru perioade și canale de comparare. De asemenea, include un USART programabil cu generator de baud rate fracționar și detecție start-of-frame, un controler/periferic SPI și un controler/periferic I²C cu mod dual. Aceste caracteristici permit comunicarea eficientă și flexibilă cu alte dispozitive și senzori.

Arduino UNO R3 integrează și un comparator analogic cu o referință de intrare scalabilă și un watchdog timer cu un oscilator pe chip separat. Aceste caracteristici contribuie la stabilitatea și securitatea funcționării plăcii în diverse aplicații. Placa oferă și șase canale PWM, care sunt esențiale pentru controlul precis al motoarelor și altor actuatori.

Un alt aspect important al Arduino UNO R3 este flexibilitatea sa în alimentare. Placa poate fi alimentată fie prin conectorul USB, fie printr-o sursă externă cu o tensiune cuprinsă între 7-12V, cu o tensiune maximă admisă pe pinul de alimentare (VIN) de 20V. Aceasta permite utilizatorilor să aleagă între diferite metode de alimentare, în funcție de cerințele proiectului lor.

Pentru a programa placa Arduino UNO R3, utilizatorii pot folosi fie Arduino IDE (offline), fie Arduino Web Editor (online). Arduino Web Editor este găzduit online și este întotdeauna actualizat cu cele mai noi funcții și suport pentru toate plăcile. [10]

3.2.3 Modulul LoRa RFM95

Modulul RFM95 este un transceiver RF de putere redusă, care utilizează modulația LoRa pentru a oferi comunicații de lungă distanță și consum redus de energie. Acest modul este ideal pentru aplicații IoT, unde sunt necesare transmisii pe distanțe mari și rezistență ridicataă la interferențe. Este utilizat frecvent în citirea automata a contoarelor, automatizarea clădirilor, monitorizarea industrial și alte aplicații de telemetrie.

În Figura 15 este prezentat cum arată acest modul LoRa RFM95.

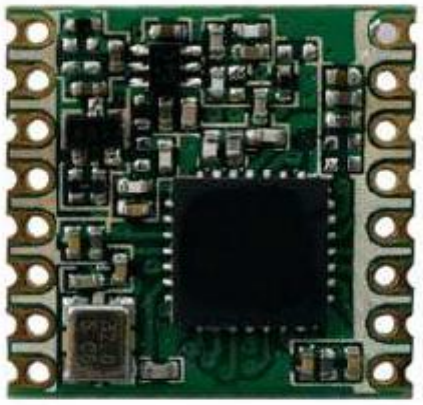

Figura 15 Modulul RFM95

RFM95 utilizează modulația LoRa pentru a atinge sensibilități de până la -148 dBm, combinată cu un amplificator de putere integrat de +20 dBm. Acest lucru oferă un buget de legătură excepțional, optim pentru aplicațiile care necesită acoperire extinsă. Modulul suportă, de asemenea, modulații FSK, GFSK, MSK și OOK pentru compatibilitate cu sistemele existente.

Modulul RFM95 este un transceiver half-duplex cu arhitectură low-IF. Semnalul RF recepționat este amplificat inițial de un amplificator de zgomot redus, urmat de conversia la componente în fază și în cusdratură la frecvența intermediară. Conversia datelor este realizată de o perehe de ADC sigma-delta, iar procesarea semnalului și demodularea sunt efectuate în domeniul digital. În Figura 16 se poate observa aceste considerente teoretice, în reprezentarea diagramei bloc.

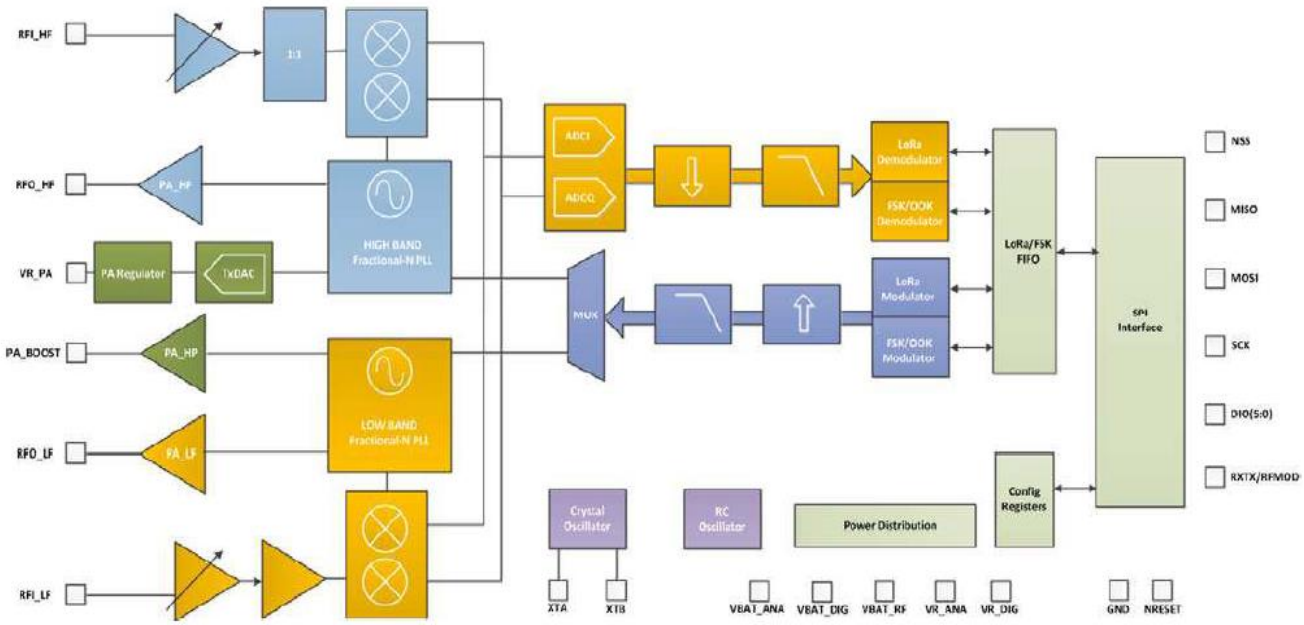

Figura 16 Diagrama bloc a modulului RFM95 [11]

În Figura 17 se poate observa diagrama pinilor modulului LoRa RFM95.

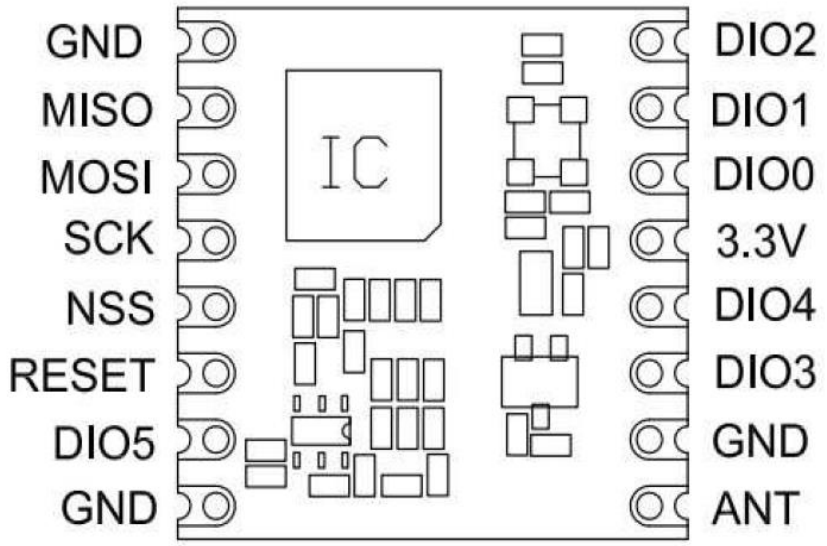

Figura 17 Diagrama pinilor modulului RFM95 [11]

În Tabelul 5 găsim descrierea pinilor modulului RFM95.

Tabelul 5 Descrierea pinilor modulului RFM95

Număr	Nume	Tip	Descriere
1	GND	-	Masă
2	MISO	I	Ieșire date SPI
3	MOSI	O	Intrare date SPI
4	SCK	I	Intrare ceas SPI
5	NSS	I	Intrare selectare cip SPI
6	RESET	I/O	Intrare declanșare resetare
7	DIO5	I/O	Intrare/Ieșire digitală, configurată software
8	GND	-	Masă
9	ANT	-	Ieșire/Ieșire semnal RF
10	GND	-	Masă
11	DIO3	I/O	Intrare/Ieșire digitală, configurată prin software
12	DIO4	I/O	Intrare/Ieșire digitală, configurată prin software
13	3.3V	-	Tensiune de alimentare
14	DIO0	I/O	Intrare/Ieșire digitală, configurată prin software
15	DIO1	I/O	Intrare/Ieșire digitală, configurată prin software
16	DIO2	I/O	Intrare/Ieșire digitală, configurată prin software

A. Caracteristici electrice

- Tensiuni de alimentare: 1.8 – 3.7 V
- Gama de temperature de operare: -20°C până la +70°C
- Consum de curent în mod Sleep: 0.2 uA
- Consum de curent în mod Recepție: 10.3mA
- Consum de curent în mod Transmisie (20 dBm): 120mA

Modulația LoRa utilizează o tehnică de spectru împrăștiat care permite o sensibilitate crescută și o rezistență superioară la interferențe. Factorul de împrăștiere și rata de corecție a erorilor sunt parametrii configurabili, permițând optimizarea performanței în funcție de aplicație.

- Factor de împrăștiere: 6 până la 12
- Lățime de bandă: 7.8 kHz până la 500 kHz
- Rată de corecție a erorilor: 4/5 până la 4/8

B. Performanțe

- Sensibilitate RF: până la -148 dBm
- Putere RF de ieșire: +20 dBm (PA_BOOST), +14 dBm (RFO_LF/HF)
- Receptoare OOK: Sensibilitate de până la -117 dBm la 4.8 kbps

RFM95 oferă o rezistență excelentă la blocare și o performanță superioară a selectivității, datorită arhitecturii avansate. De asemnea, include un detector de baterie scăzută și un senzor de temperatură integrat.

Configurarea modului se face printr-o interfață SPI care oferă acces la registrele de configurare. Aceasta include un secvențiator automat de moduri care gestionează tranzițiile și calibrarea între modurile de operare.

RFM95 operează în benzile de frecvență ISM fără licență, inclusiv:

- 433 MHz
- 868 MHz
- 915 MHz

Aceste frecvențe permit utilizarea modulului în diverse regiuni și aplicații, respectând reglementările locale

Modulul RFM95 suportă mai multe moduri de operare, incluzând:

- Modul de standby: Consum redus de energie în perioadele inactive.
- Modul de recepție continuă: Monitorizează constant semnalele RF.

- Modul de transmisie: Transmite date cu putere variabilă
- Modul de economisire a energiei: Configurații specific pentru minimizarea consumului.

Modulul include funcționalități avansate de securitate, precum :

- Criptare AES-128: Asigură confidențialitatea datelor transmise
- Autentificare: Verifică identitatea dispozitivelor pentru prevenirea accesului neautorizat

Integrarea modulului RFM95 într-un proiect presupune conectarea acestuia la un microcontroler compatibil și utilizarea unei biblioteci software, care oferă funcționalitățile necesare pentru comunicarea LoRa. Configurarea parametrilor de tranmisie, cum ar fi frecvența, lățimea de bandă și factorul de împrăștiere, se face prin setările din registrele modulului.

Modemul LoRa utilizează două tipuri de format: explicit și implicit. Pachetul explicit include un antet scurt care conține informații despre numărul de octeți, rata de codare și dacă pachetul utilizează CRC. Formatul pachetului este ilustrat in Figura 18.

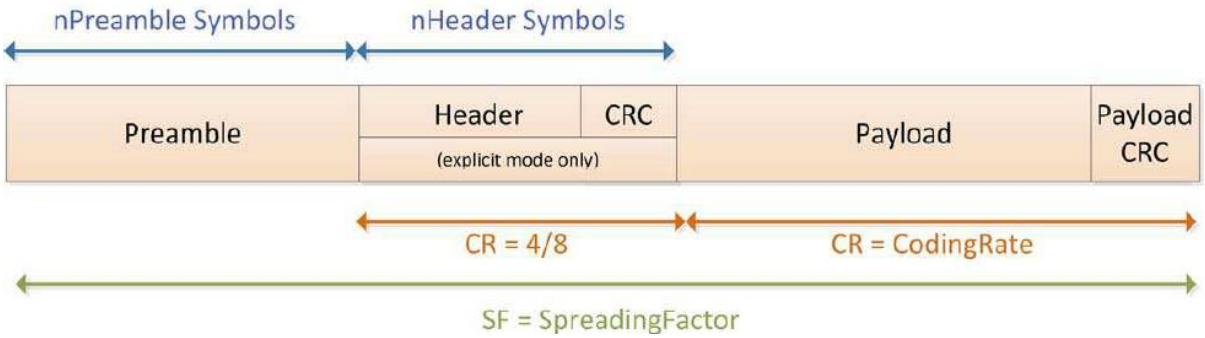

Figura 18 Structura unui pachet LoRa [11]

În modul de recepție unic, modemul caută un preambul într-o fereastră de timp prestabilită. Dacă nu se detectează un preambul la sfârșitul ferestrei, se generează o întrerupere RxTimeout, iar cipul revine în modul stand-by. La recepția unui pachet, se generează întrerupere RxDone și, dacă CRC-ul este valid, datele sunt scrise în buffer-ul FIFO.

În modul de recepție continuă, modemul scanează canalul pentru un preambul. După detectarea unui preambul, modemul îl urmărește până la primirea pachetului și continuă să aștepte următorul preambul. Spre deosebire de modul de recepție unic, în acest mod dispozitivul nu revine în stand-by la generarea unei întreruperi de timeout.

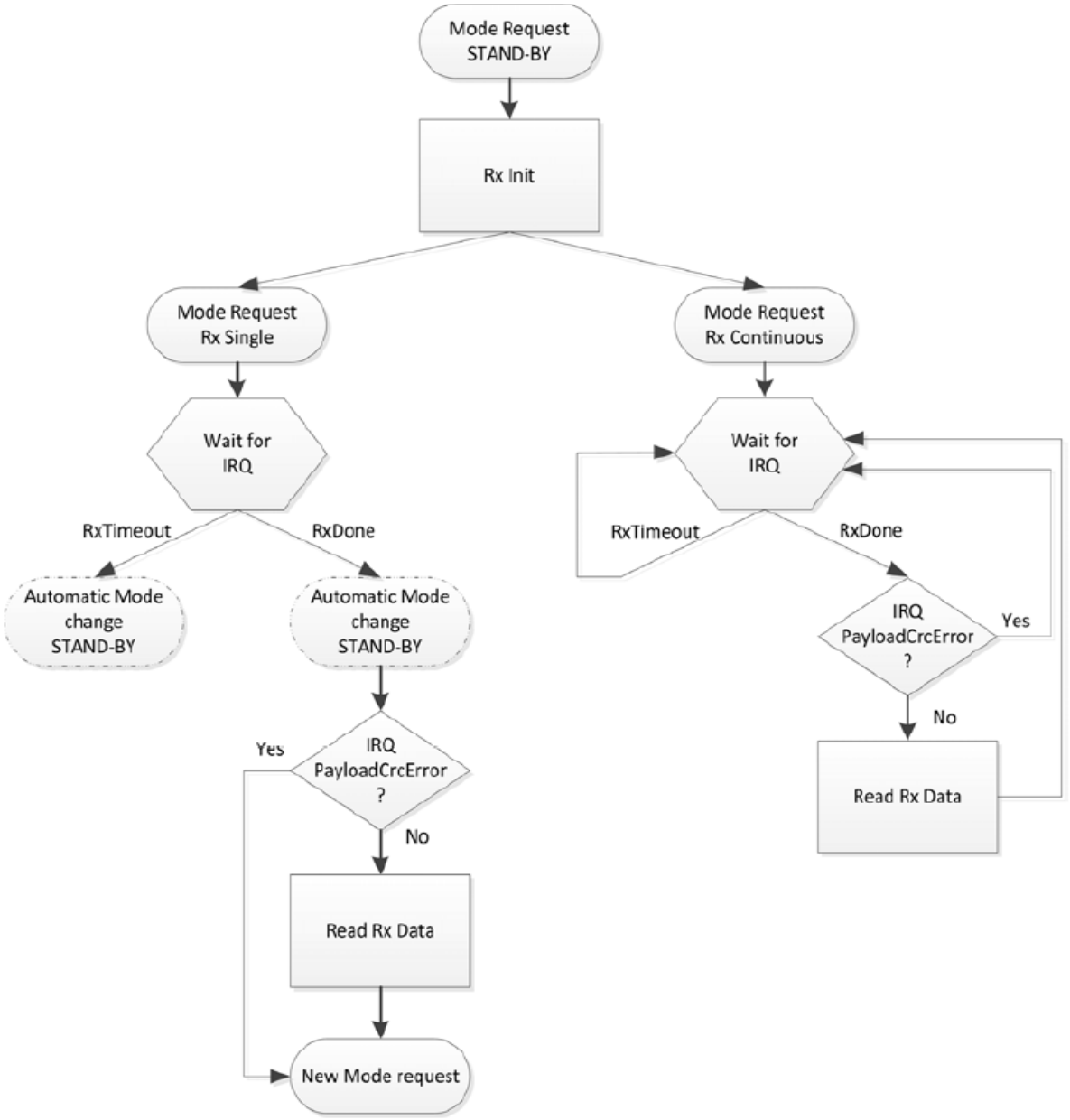

Figura 19 Secvența de recepție LoRa [11]

În modul de transmisie, consumul de energie este oprimizat prin activarea blocurilor RF, PLL și PA doar atunci când este necesară transmiterea datelor pachetului. Secvența tipică de transmisie LoRa implică configurarea registrului FifoPtrAddr la baza FIFO de transmisie și scrierea datelor pachetului în FIFO. Transmisia este inițială prin trimiterea unei comenzi de mod TX. La finalizarea transmisiei, se generează întreruperea TxDone, iar radioul revine în modul stand-by.

Rata simbolurilor (Rs) este determinată de lățimea de bandă programată (BW) și de factorul de răspândire (SF). Aceasta este dată de formula:

$$R_s = \frac{BW}{2^{SF}} \quad (4)$$

Acest parametru este esențial pentru determinarea eficienței transmisiei și a ratei de date a semnalului LoRa.

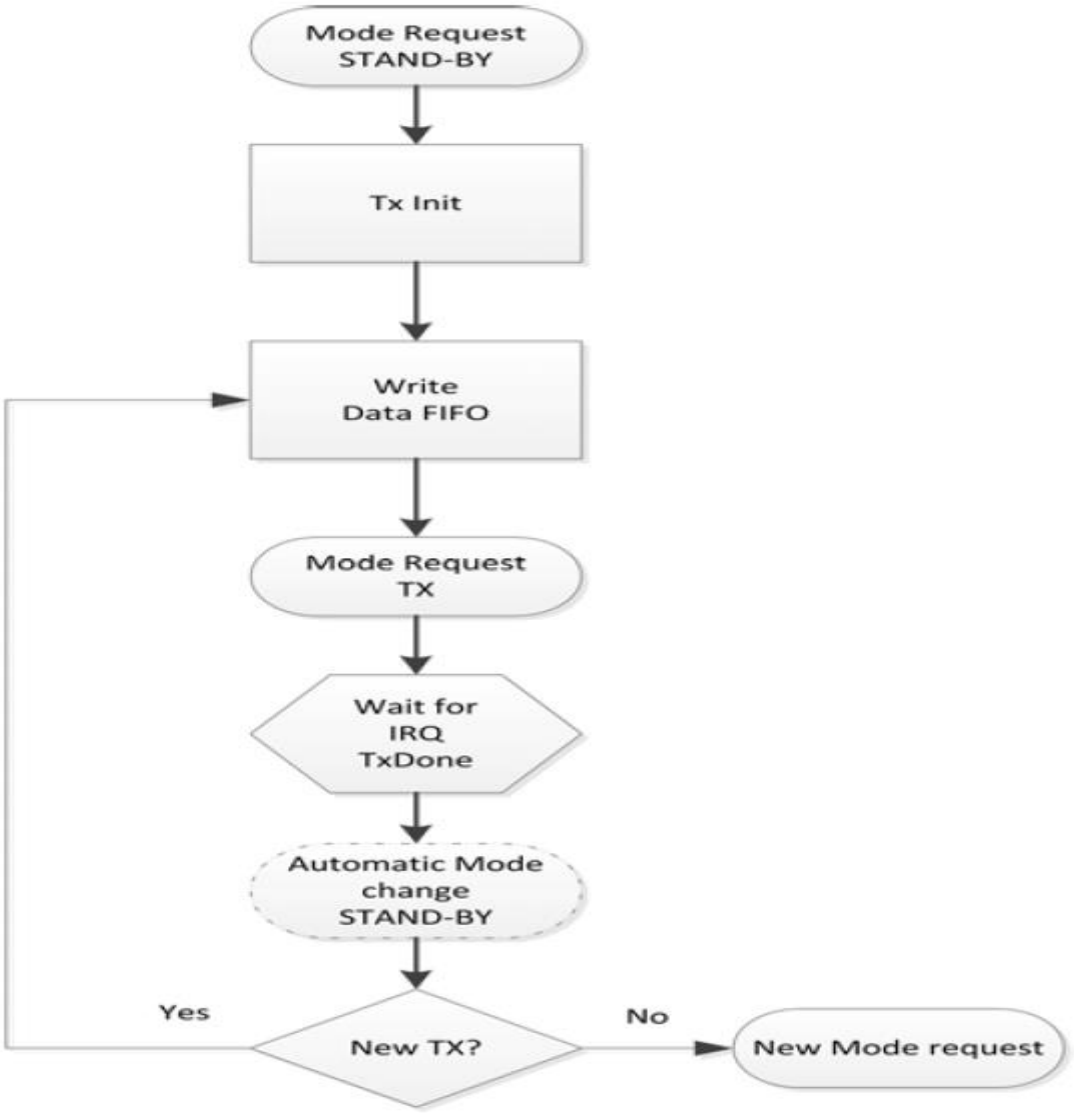

Figura 20 Secvența de transmisie a LoRa [11]

Modulul LoRa RFM95 oferă o combinație excepțională de acoperire extinsă, consum reuds de energie și rezistență la interferențe. Acesta este un component esențial pentru dezvoltarea sistemelor de comunicație de lungă distanță, oferind flexibilitatea și performanța necesare pentru aplicații moderne și complexe. [11]

3.2.4 Senzorul cu ultrasunete HC-SR04

Senzorii ultrasonici folosesc unde sonore pentru a determina distanța dintre senzor și cel mai apropiat obiect din calea sa. Acești senzori operează la frecvențe mai mari decât cele percepute de urechea umană. Principiul de funcționare se bazează pe trimiterea unui impuls

sonor și măsurarea timpului necesar pentru ca ecoul să revină la senzor. Formula de bază pentru calcularea distanței este:

$$d = v \times t \tag{5}$$

Unde:

- d reprezintă distanța
- v este viteza sunetului (aproximativ 343m/s la condiții și presiune standard)
- t este timpul de călătorie al undei sonore.

Senzorul HC-SR04 este unul dintre cei mai populari senzori ultrasonici datorită costului redus, greutății mici și suportului extins. Specificațiile tehnice ale HC-SR04 sunt esențiale pentru înțelegera performanței și a modului în care poate fi integrat în proiecte de integrat în proiecte. Aceste specificații includ:

- Alimentare: +5V DC
- Curent inactiv: <2 mA
- Curent de lucru: 15 mA
- Unghi efectiv: <15°
- Distanță de măsurare: 2-400 cm
- Rezoluție: 0.3 cm
- Unghi de măsurare: 30°
- Lățimea impulsului de intrare TRIG: 10 µs
- Dimensiuni: 45mm x 20mm x 15mm
- Greutate: aproximativ 10 g

În Figura 21 se ilustrează senzorii ultrasonici HC-SR04.

Figura 21 Senzorii ultrasonici HC-SR04 [13]

Senzorul HC-SR04 funcționează prin trimiterea unui impuls sonor și măsurarea timpului necesar pentru ca ecoul să revină la senzor. Procesul de măsurare a distanței implică mai mulți pași:

1. Setarea pinului TRIG la HIGH pentru 10 microsecunde.
2. Senzorul trimite un impuls sonor de opt cicluri la 40 KHz.
3. Pinul ECHO devine HIGH și rămâne astfel până când ecoul revine la senzor.
4. Măsurarea rimpului în care pinul ECHO rămâne HIGH.
5. Calcularea distanței folosind formula

$$d = \frac{v \times t}{2} \quad (6)$$

Pentru a transforma timpul măsurat în microsecunde într-o distanță în centrimetri, se folosește formula

$$d = \frac{t}{58} \quad (7)$$

Unde :

- d este distanța în centimetri
- t este timpul în microsecunde

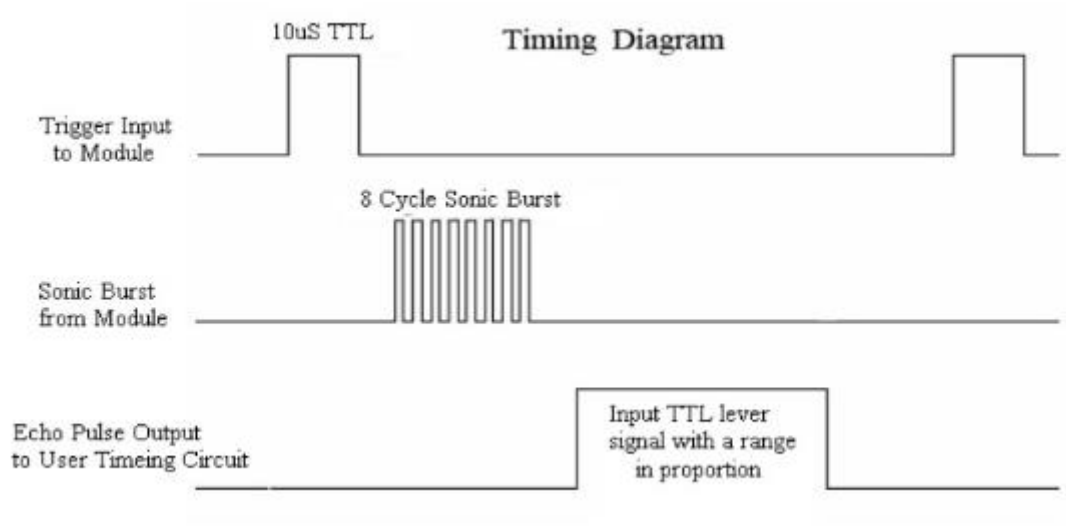

Figura 22 Diagrama de timp a HC-SR04

Senzorul HC-SR04 are patru pini: VCC, GND, TRIG, ECHO. Conectarea senzorlui la un microcontroler implică următorii pași:

1. VCC: Se conectează la sursa de alimentare de 5V a microcontrolerului.
2. GND: Se conectează la masa microcontroler-ului.
3. TRIG: Se conectează la un GPIO care poate fi setat la HIGH pentru a declanșa impulsul sonic.
4. ECHO: Se conectează la un pin GPIO pentru a citi durata impulsului.

Deși senzorii ultrasonici sunt foarte utili pentru mai multe aplicații, aceștia pot prezenta unele limitări. Printre principalele surse de erori se numără:

- Ecosuri neintenționate: În spațiile închise, sunetul poate reflecta de pe mai multe suprafețe, cauzând măsurători incorecte
- Suprafețe neuniforme: Dacă obiectul nu prezintă o suprafață plată, undele sonore pot fi deviate, rezultând în măsurători eronate.
- Limitările de viteză: Senzorii ultrasonici au o limitare în ceea ce privește viteza de măsurare a distanței, în special la distanțe mai mari.

Pentru a reduce numărul de erori, se pot folosi diverse metode de protecție și calibrare, precum și reducerea frecvenței de trimitere a impulsurilor pentru a permite dispersarea ecourilor anterioare. [12]

3.2.5 Divizorul de tensiune

Divizorul de tensiune este un circuit esențial în electronică, folosit pentru a reduce tensiunea de intrare la un nivel mai mic, adecvat pentru diverse aplicații. Funcționarea sa se bazează pe legea lui Ohm și pe distribuția tensiunii într-un circuit serie format din două sau mai multe rezistențe.

Un divizor de tensiune simplu constă din două rezistențe R_1 și R_2 conectate în serie. Tensiunea de ieșire V_{out} este preluată de la punctul comun dintre cele două rezistențe. Tensiunea aplicată la capetele seriei de rezistențe este V_{in}. Relația matematică care descrie tensiunea de ieșire este:

$$Vout = Vin \times \frac{R2}{R1+R2} \quad (8)$$

unde:

- V_{in} reprezintă tensiunea de intrare.
- V_{out} reprezintă tensiunea de ieșire.
- R_1 și R_2 sunt valorile rezistențelor din circuit.

Această formulă se bazează pe legea lui Ohm și pe împărțirea tensiunii într-un circuit serie. [14]

Pentru a ilustra conceptul, în Figura 23 este prezentată schema de bază a unui divizor de tensiune format din două rezistențe, R_1 și R_2. Tensiunea de intrare $V_{intrare}$ este aplicată la capetele seriei de rezistențe, iar tensiunea de ieșire $V_{ieșire}$ este preluată de la punctul comun dintre cele două rezistențe.

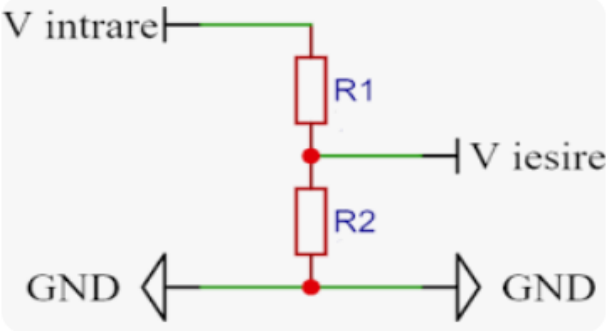

Figura 23 Schema unui divizor de tensiune standard [15]

Un avantaj major al divizorului de tensiune este simplitatea și costul redus al componentelor necesare. Cu toate acestea, există și anumite limitări: divizoarele de tensiune sunt potrivite doar pentru aplicații cu curenți mici, deoarece curenții mari pot cauza deviații semnificative în tensiunea de ieșire și pot genera căldură excesivă în rezistențe.

4 MEDII DE DEZVOLTARE SOFTWARE FOLOSITE

Arduino IDE
Python
InfluxDB

Pentru realizarea acestui proiect au fost utilizate mai multe medii de dezvoltare software, fiecare având un rol specific în implementarea și funcționarea sistemului. Aceste medii includ Arduino IDE pentru programarea microcontrolerelor, Python pentru prelucrarea datelor și InfluxDB pentru stocarea și vizualizarea datelor.

4.1 ARDUINO IDE

Arduino Integrated Development Environment (IDE) este un mediu de dezvoltare open-source utilizat pentru scrierea și încărcarea codului pe plăci Arduino. Acesta oferă un editor de cod simplu, dar eficient, și un set de biblioteci predefinite care facilitează interacțiunea cu diverse componente hardware. [16]

Caracteristici tehnice ale Arduino IDE:

- Editor de cod simplu: Permite scrierea și editarea codului în limbajul de programare C/C++ specific Arduino.
- Upload ușor: Permite încărcarea rapidă a codului pe plăcile Arduino prin intermediul unui cablu USB.
- Biblioteci predefinite: Oferă suport pentru senzori, module de comunicație și alte componente hardware prin intermediul unor biblioteci standardizate.
- Monitor serial: Permite vizualizarea în timp real a datelor trimise de microcontroler către computer, facilitând debug-ul și testarea codului.

4.2 PYTHON

Python este un limbaj de programare interpretat, de nivel înalt, cunoscut pentru sintaxa sa clară și concisă. În acest proiect, Python a fost utilizat pentru prelucrarea datelor recepționate de la sistemul de senzori și pentru comunicarea cu baza de date InfluxDB.

Caracteristici tehnice ale Python:

- Sintaxă clară și concisă: Facilitează scrierea și citirea codului.

- Biblioteci extinse: Python dispune de o gamă largă de biblioteci, care simplifică dezvoltarea aplicațiilor complexe.
- Interoperabilitate: Python poate fi utilizat pentru a interacționa cu diverse sisteme și platforme, inclusive baze de date, API-uri web și dispositive hardware. [17]

4.3 INFLUXDB

InfluxDB este o bază de date open-source optimizată pentru stocarea și interogarea seriilor temporale de date. Aceasta este ideală pentru aplicații care necesită monitorizarea și analizarea datelor în timp real.

Caracteristici tehnice ale InfluxDB:

- Stocarea seriilor temporale: Permite stocarea eficientă a datelor structurate în funcție de timp, cum ar fi măsurători de senzori sau date de monitorizare.
- Interogări rapide: Oferă un limbaj de interogare specializat (Flux) pentru extragerea rapidă și eficientă a datelor.
- Vizualizare: Integrează un instrument de vizualizare a datelor care permite crearea de grafice și dashboard-uri personalizate. [18]

5 PROIECTAREA SISTEMULUI PROPUS

5.1 PROPUNERE DE SISTEM

Proiectul propus vizează cartografierea spațiilor inaccesibile folosind un system de monitorizare bazat pe tehnologia de transmisie LoRa la frecvența de 868 MHz. Sistemul utilizează un senzor HC-SR04 pentru măsurarea distanțelor, conectat la un microcontroler Arduino pentru transmisie și recepție.

Datele colectate de senzor sunt transmise fără fir de la unitatea de transmisie la unitatea de recepție folosind module LoRa RFM95. Aceste date sunt stocate și vizualizate utilizând InfluxDB, o bază de date optimizată pentru datele de tip serie temporală. Vizualizarea datelor se realizează direct în InfluxDB, facilitând analiza și interpretarea acestora.

Pentru dezvoltarea și implementarea acestui sistem, se utilizează Arduino IDE pentru programarea microcontrolerelor și Python pentru prelucrarea datelor și trimiterea acestora către InfluxDB. Acest proiect demonstrează integrarea eficientă a componentelor hardware și software pentru monitorizare și cartografiere.

5.2 PROIECTARE HARDWARE

În acest subcapitol, este detaliat procesul de proiectare hardware al proietului „Sistem de cartografiere spații inaccesibile și transmisia prin LoRa”, care se bazează pe transmiterea și recepționarea datelor, de la un senzor, prin intermediul comunicației LoRa. Proiectul este împărțit în două părți principale: partea de transmisie și partea de recepție.

5.2.1 Transmisia

Partea de transmisie a proiectului este instalată pe o mașinuță cu telecomandă, concepută pentru a cartografia spațiile inaccesibile. Utilizând senzorul HC-SR04, datele colectate sunt transmise prin modulul LoRa RFM95 către stația de recepție. Această configurație permite explorarea zonelor dificile, cum ar fi peșterile, fără a necesita prezența umană.

Cartografierea este realizată în plan datorită utilizării unei singure perechi de senzori ultrasonici, montați lateral pe mașinuță. Această dispunere permite măsurarea precisă a pereților peșterii pe măsură ce mașinuța se deplasează.

În Figura 24 este prezentată schema electrică a părții de transmisie a sistemului. Această schemă ilustrează modul în care sunt conectate componenetele: senzorul HC-SR04, microcontrlerul Arduino UNO R4 Minima și modulul de comunicație LoRa RFM95.

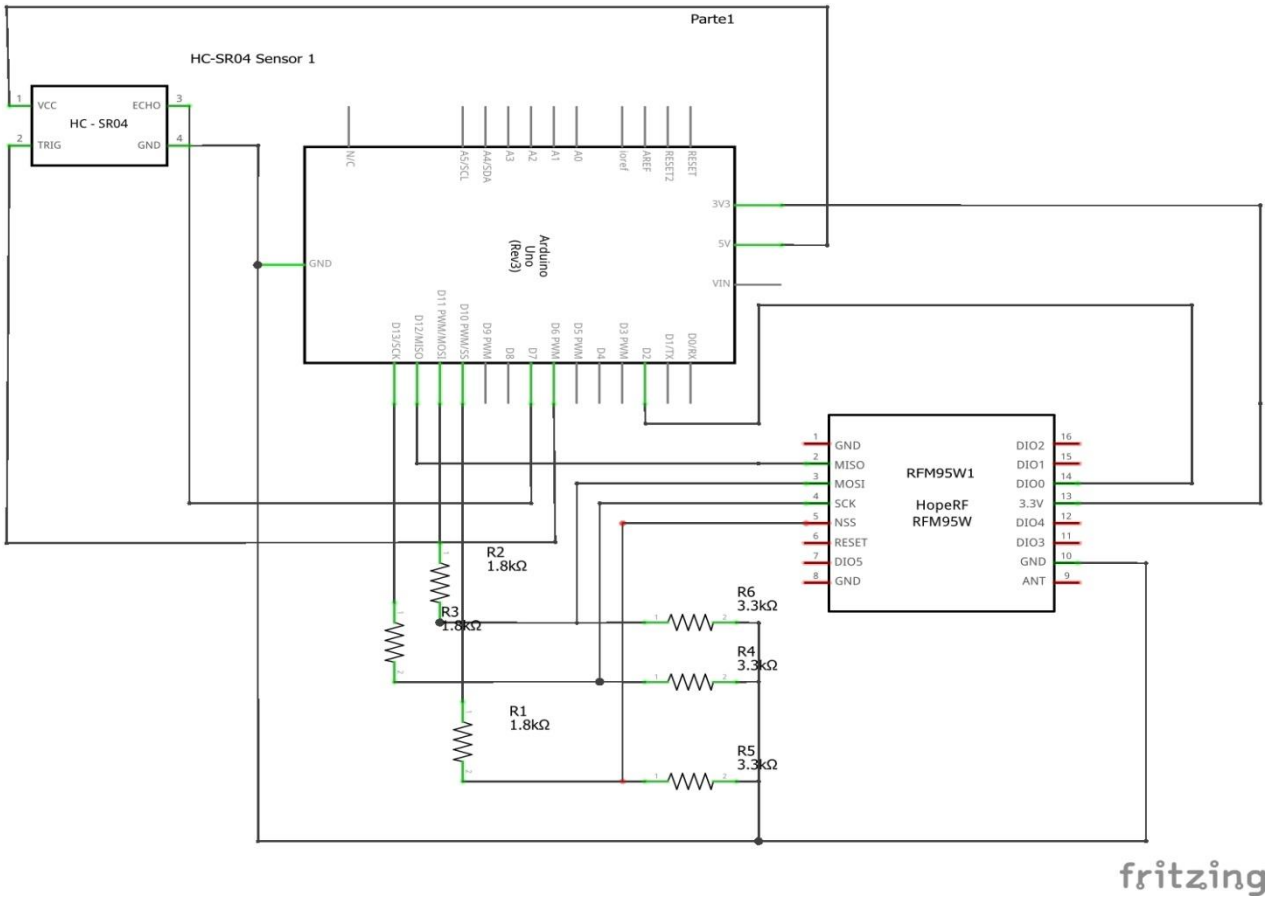

Figura 24 Schema electrică a transmisiei

Pentru a utiliza modulul LoRa RFM95, au fost necesare câteva ajustări hardware suplimentare deoarece acest modul nu vine cu pini standard. Astfel, am lipit modulul RFM95 pe o placă adaptoare ESP8266, ceea ce a permis conectarea la Arduino UNO R4 Minima.

Un aspect important al configurării hardware este conectarea unei antene la pinul ANA al modulului RFM95. Antena este esențială în transmiterea și recepționrea semnalului radio. Am folosit un fir de cupru pentru a crea această antenna, carea a fost ajustată la o lungime specifică pentru a optimiza performanța la frecvența de 868 MHz. Calculul lungimii antenei se bazează pe formula pentru lungimea de undă:

$$L = \frac{c}{f} \tag{9}$$

Unde:

- L este lungimea de undă,
- c este viteza luminii (aproximativ 300.000.000 m/s),
- f este frecvența (868.000.000 Hz pentru frecvența de 868MHz).

Aplicând formula 9 :

$L = \frac{300.000.000}{868.000.000} = 0.345\ m$

Pentru a obține o antenă eficientă, am folosit un sfert din lungimea de undă calculată, deoarece antenele tip sfert de undă sunt frecvent utilizate în aplicații de radio frecvență datorită dimensiunii compacte și performanței adecvate. Astfel, lungimea antenei a fost calculată ca fiind:

$$\frac{0.345}{4} = 0.086\ m = 8.6\ cm$$

Prin urmare, antena a fost tăiată la 8.6 cm pentru a corespunde optim frecvenței de 868 MHz. Această ajustare asigură că semnalul radio este transmis și recepționat eficient.

Senzorii HC-SR04 sunt conectați la placa Arduino UNO pentru a măsura distanțele față de obstacole. Fiecare senzor are patru pini: VCC, Trig, Echo și GND. Pinul VCC este conectat la linia de alimentare de 5V a plăcii Arduino, iar pinul GND la masa plăcii (GND). Pinul Trig este conectat la pinul digital D6 al plăcii Arduino și inițiază impulsurile ultrasonice. Pinul Echo, conectat la pinul digital D7, primește semnalul reflectat de la obstacol și măsoară timpul necesar pentru ca impulsul să revină.

Conexiunile dintre modulul LoRa RFM95 și Arduino UNO sunt esențiale pentru a asigura o comunicare eficientă între aceste două componente, având în vedere diferențele de tensiune dintre pinii SPI ai Arduino (5V) și cei ai RFM95 (3.3V). Pentru a proteja modulul RFM95 și a-l face compatibil cu Arduino, am utilizat divizoare de tensiune.

Divizorul de tensiune este format din rezistențele de 1.8 kΩ și 3.3 kΩ, iar calculul acestuia este următorul:

$$Vout = Vin \times \frac{R2}{(R1 + R2)} = 5V \times \frac{3.3k}{(1.8k + 3.3k)} = 3.24V$$

Pinul MISO al modulului RFM95 este conectat direct la pinul digital 12 al Arduino, deoarece MISO este un pin de ieșire și poate suporta tensiunea de 5V. Pinul MOSI al modulului RFM95 este conectat la pinul digital 11 al Arduino printr-un divizor de tensiune, care reduce

tensiunea de la 5V la 3.3V. Aceeași metodă de reducere a tensiunii este utilizată și pentru pinul SCK, care este conectat la pinul digital 13 al Arduino printr-un divizor de tensiune similar. Pinul NSS al modulului RFM95 este conectat la pinul digital 10 al Arduino, tot printr-un divizor de tensiune pentru a adapta tensiunea la 3.3V. Pinul DIO0 al modulului RFM95 este conectat direct la pinul digital 2 al Arduino, deoarece acest pin de semnal poate funcționa corect la tensiunea de 5V fără a necesita un divizor de tensiune. Pinul GND al modulului RFM95 este conectat la pinul GND al Arduino pe breadboard. Alimentarea modulului RFM95 se face prin conectarea pinului 3.3V al acestuia la pinul 3.3V al Arduino.

În Figura 25 este prezentată schema de cablare a sistemului de transmisie, ilustrând modul în care componentele sunt conectate pe o placă de testare (breadboard).

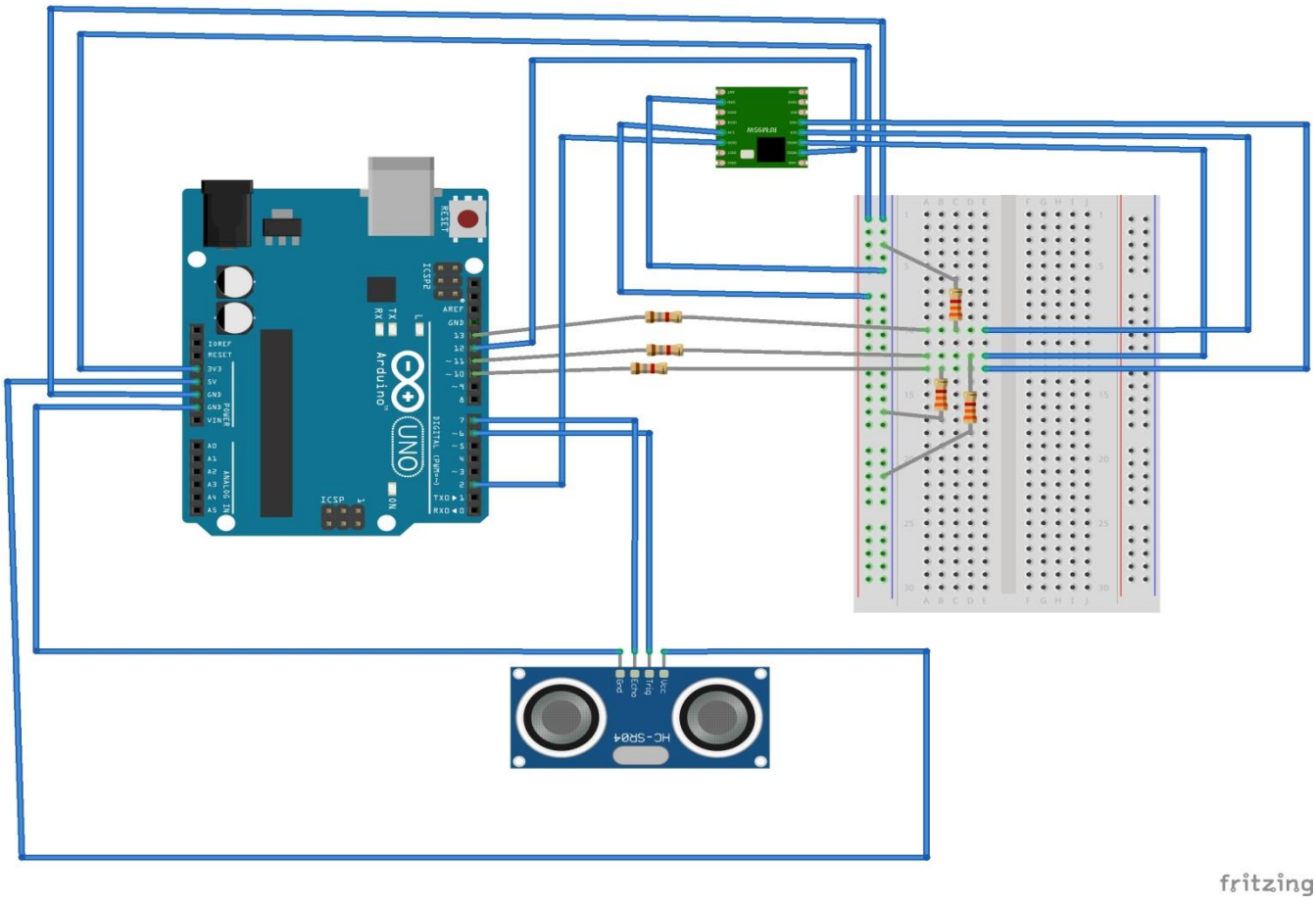

Figura 25 Schema de cablare pentru transmisie

Codul necesar pentru funcționarea sistemului va fi încărcat pe placa Arduino, iar alimentarea sistemului de transmisie se va realiza prin intermediul unei baterii alcaline de 9V care este conectată prin intermediul unui soclu care are conectat minusul la GND al Arduino și plusul la pinul V_{in}, asigurând funcționarea autonomă a acestuia.

5.2.2 Recepția

Partea de recepție asigură preluarea datelor transmise de la sistemul de transmisie și trimiterea acestora către un laptop pentru stocare și analiză.

În Figura 26 este prezentată schema electrică a sistemului de recepție, ilustrând conexiunile dintre modulul RFM95 și Arduino UNO R3.

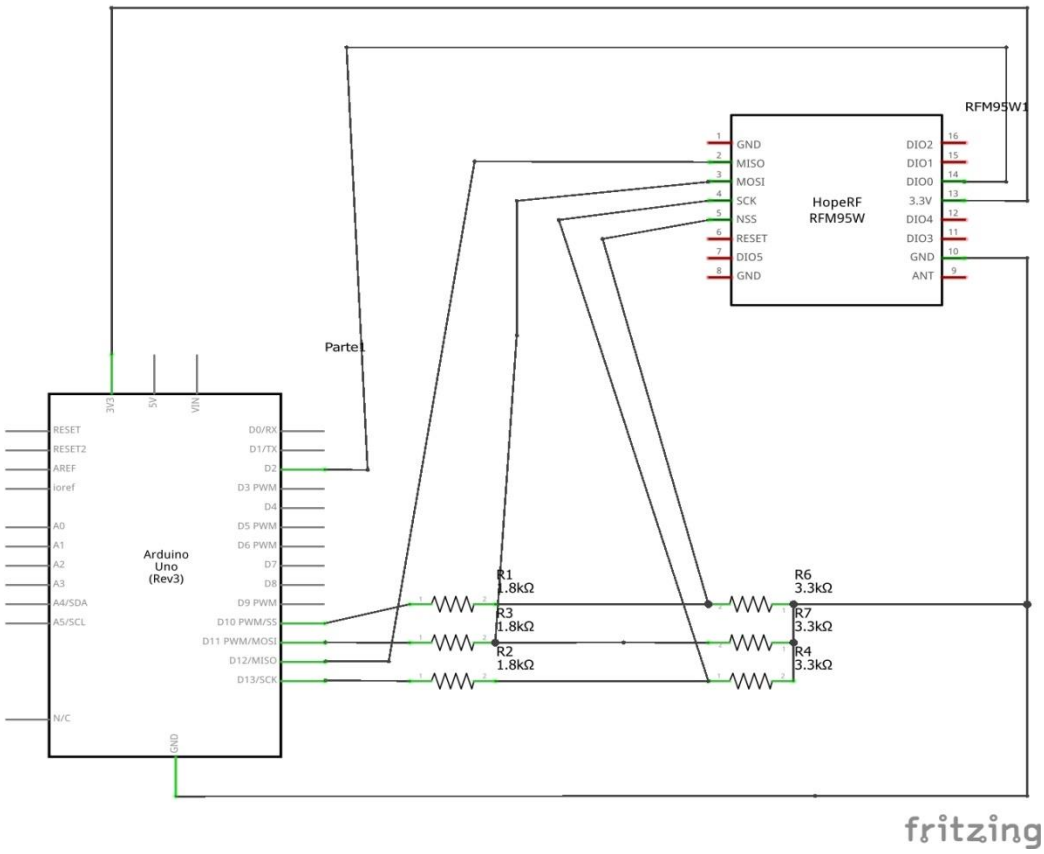

Figura 26 Schema electrică a recepției

Pentru a realiza conexiunile necesare, a fost necesar să lipesc câte un fir jumper la fiecare pin al modulului RFM95. Acest lucru a permis o conectare sigură la placa Arduino. La fel ca în partea de transmisie, am utilizat o antenă realizată dintr-un fir de cupru, tăiat la lungimea de 8.6 cm, optimizat pentru frecvența de 868 MHz utilizând formula 9.

La partea de recepție, conexiunile dintre modulul RFM95 și placa Arduino UNO R3 sunt realizate într-un mod similar cu partea de transmisie, folosind divizoare de tensiune pentru a adapta nivelurile de tensiune de 5V și 3.3V acolo unde este necesar. Divizoarele sunt realizate cu rezistențe de 1.8 kΩ și 3.3kΩ, conform formulei 8.

Pinul MISO al modulului RFM95 este conectat direct la pinul digital 12 al Arduino. Pinul MOSI este conectat la pinul digital 11. Pinul SCK al modulului este conectat la pinul digital 13, iar pinul NSS este conectat la pinul digital 10. Pinii MOSI, SCK și NSS sunt conectați prin divizoare

de tensiune pentru a asigura funcționarea corectă la tensiunea de 3.3V. Pinul DIO0 al modulului RFM95 este conectat direct la pinul digital 2 al Arduino deoarece poate funcționa corect la tensiunea de 5V. Pinii de alimentare GND și 3.3V ai modulului RFM95 sunt conectați la pinii corespunzători Arduino.

În Figura 27 este prezentată schema de cablare pentru partea de recepție a sistemului, ilustrând conexiunile între modulul RFM95, Arduino UNO și rezistențele utilizate pentru divizoarele de tensiune.

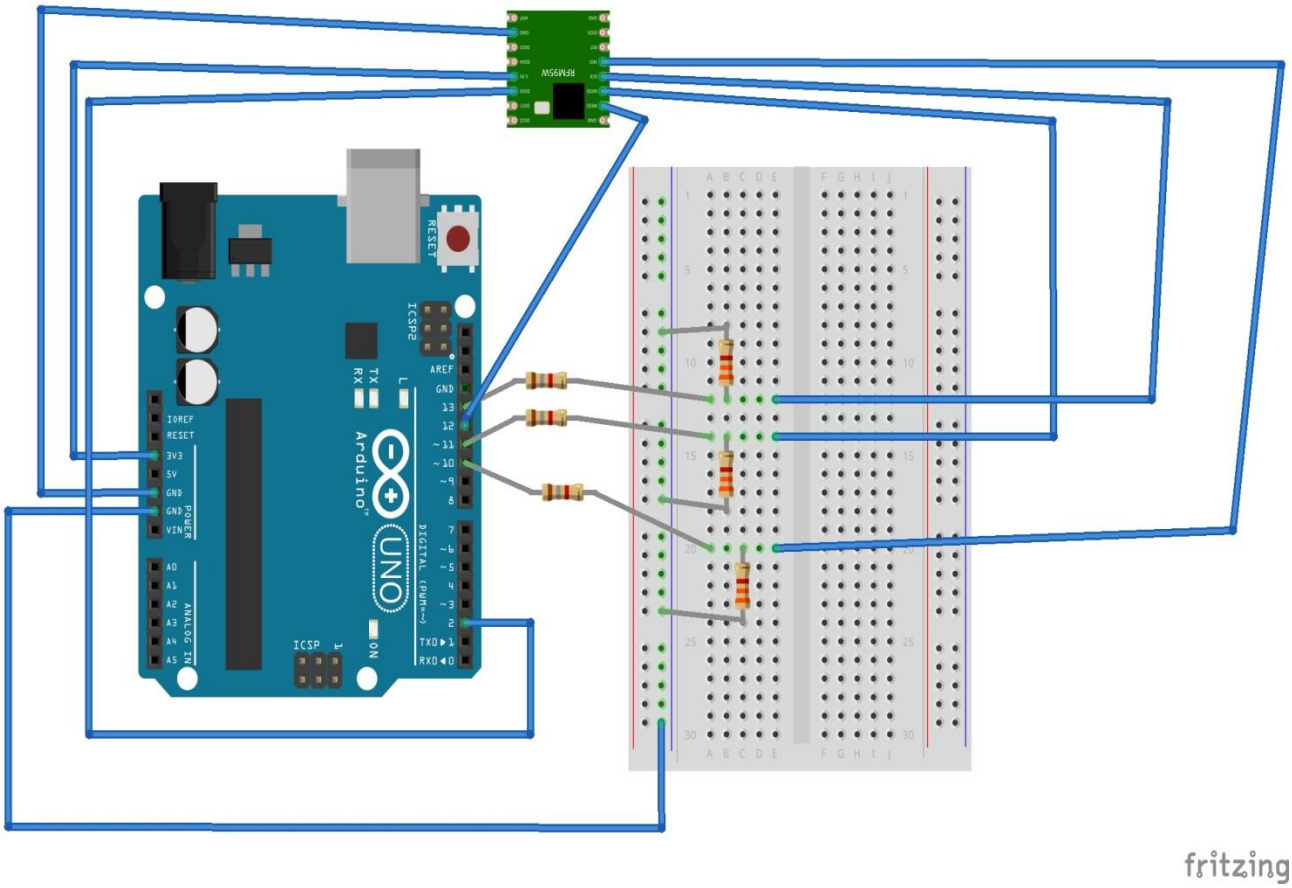

Figura 27 Schema de cablare pentru recepție

Partea de recepție a sistemului preia datele primite de la modulul LoRa RFM95 și le transmite către laptop prin intermediul unui cablu USB. Pentru a asigura această funcționalitate, se utilizează coduri specifice, care sunt încărcate pe placa Arduino UNO R3, gestionând atât comunicarea cu modulul LoRa, cât și transferul de date către laptop.

5.3 PROIECTARE SOFTWARE

Proiectarea software a acestui proiect implică dezvoltarea codului pentru gestionarea comunicării și procesării datelor între senzori, modulele LoRa și baza de date InfluxDB. Codul este structurat în două părți principale: codul de transmisie și recepție pentru Arduino și codul Python pentru preluarea și stocarea datelor în InfluxDB.

5.3.1 Arduino IDE

Pentru a realiza proiectul, am inclus două librării esențiale: SPI.h și LoRa.h.

```
1   #include <SPI.h>  // Librărie pentru comunicarea SPI
2   #include <LoRa.h> // Librărie pentru comunicarea LoRa
```

Figura 28 Librării folosite în Arduino IDE

Librăria SPI.h este folosită pentru a permite comunicarea prin protocolul SPI, necesară pentru interacțiunea cu diverse module. Librăria LoRa.h este specifică pentru modulele LoRa, utilizate pentru transmisii de date pe distanțe lungi, asigurând inițializarea și gestionarea comunicării LoRa. Aceste librării sunt folosite atât pentru partea de transmisie cât și pentru partea de recepție. De asemenea, am folosit atât la codul de transmisie cât și la recepție ('Serial.begin(9600)'), asigurând astfel că datele pot fi transmise între Arduino și computer la o viteză adecvată. Linia 'while (!Serial);' așteptă deschiderea conexiunii seriale, garantând că inițializarea continuă doar după ce conexiunea este stabilită, urmând să se afișeze un mesaj pentru a indica procesul de inițializare.

```
void setup() {
  Serial.begin(9600);  // Inițializează comunicarea serială la 9600 baud rate
  while (!Serial);  // Așteaptă deschiderea conexiunii seriale

  Serial.println("Inițializare LoRa...");
```

Figura 29 Inițializare comunicare serială

- Transmisie

În Figura 30 este prezentată o secțiune a codului în care sunt definiți pinii utilizați pentru comunicarea cu modulul LoRa și senzorul ultrasonic.

```
4   #define NSS 10   // Pinul de selectare a modulului LoRa
5   #define DIO0 2   // Pinul DIO0 al modulului LoRa
6   #define MOSI_PIN 11 // Pinul MOSI (Master Out Slave In) pentru SPI
7   #define MISO_PIN 12 // Pinul MISO (Master In Slave Out) pentru SPI
8   #define SCK_PIN 13  // Pinul SCK (Serial Clock) pentru SPI
9
10  #define TRIG_PIN 6  // Pinul TRIG al senzorului ultrasonic
11  #define ECHO_PIN 7  // Pinul ECHO al senzorului ultrasonic
```

Figura 30 Definire pini transmisie

A urmat să configurez pinii necesari pentru modulul LoRa și senzorul ultrasonic. Pinii NSS și MOSI sunt setați ca ieșiri și inițializați la HIGH pentru a asigura selectarea corectă a dispozitivelor. Pinul DIO0 este configurat ca intrare pentru a recepționa semnalele de la modulul LoRa. Pinii TRIG și ECHO sunt configurați pentru a controla și citi datele de la senzorul ultrasonic. Pentru a începe comunicația SPI inițializarea acesteia este necesară.

```
19    // Configurare pinii ca ieșire
20    pinMode(NSS, OUTPUT);
21    digitalWrite(NSS, HIGH);
22
23    pinMode(DIO0, INPUT);
24
25    // Configurare pini SPI
26    pinMode(MOSI_PIN, OUTPUT);
27    digitalWrite(MOSI_PIN, HIGH);
28
29    pinMode(SCK_PIN, OUTPUT);
30    digitalWrite(SCK_PIN, HIGH);
31
32    pinMode(MISO_PIN, INPUT);
33
34    // Configurare pinii senzorului ultrasonic
35    pinMode(TRIG_PIN, OUTPUT);
36    pinMode(ECHO_PIN, INPUT);
37
38    // Inițializare SPI
39    SPI.begin();
```

Figura 31 Configurație pini transmisie

Pentru a putea comunica wireless, a trebuit să inițializez modulul LoRa la frecvența de 868 MHz. În caz de eșec, se afișează un mesaj de eroare.

```
41    // Inițializare LoRa
42    if (!LoRa.begin(868E6)) {  // Setează frecvența de 868 MHz
43      Serial.println("Inițializarea LoRa a eșuat!");
44      while (1);  // Blochează programul în cazul unei erori
45    }
46    Serial.println("LoRa inițializat!");
47  }
```

Figura 32 Inițializare LoRa

În continuare, a trebuit sa trimit un puls către senzorul ultrasonic pentru a declanșa măsurarea distanței. După trimiterea pulsului, am citit durata pulsului reflectat și am calculat distanța în centimetri.

```
49  void loop() {
50    long duration, distance;
51
52    // Trimitere puls TRIG
53    digitalWrite(TRIG_PIN, LOW);
54    delayMicroseconds(2);
55    digitalWrite(TRIG_PIN, HIGH);
56    delayMicroseconds(10);
57    digitalWrite(TRIG_PIN, LOW);
58
59    // Citire durata pulsului ECHO
60    duration = pulseIn(ECHO_PIN, HIGH);
61    distance = (duration / 2) / 29.1;  // Calculare distanță în cm
```

Figura 33 Măsurare distanță cu senzorul ultrasonic

În final, am configurat trimiterea datelor măsurate prin LoRa, utilizând comenzile pentru inițierea și finalizarea pachetului, urmate de un delay de 1 secundă.

```
63    // Trimitere mesaj LoRa
64    Serial.print("Distanța transmisă: ");
65    Serial.println(distance);
66    LoRa.beginPacket();
67    LoRa.print(distance);
68    LoRa.endPacket();
69
70    delay(1000); // Delay de 1 secundă
71  }
```

Figura 34 Transmiterea datelor măsurate

- Recepție

Precum la transmisie, și la recepție am început cu definirea pinilor necesari pentru comunicarea cu modulul LoRa.

```
4   #define NSS 10   // Pinul de selectare a modulului LoRa
5   #define DIO0 2   // Pinul DIO0 al modulului LoRa
```

Figura 35 Definire pini recepție

În continuare, am setat pinii necesari pentru modulul LoRa folosind funcția 'LoRa.setPins(NSS, -1, DIO0)' , unde NSS și DIO0 sunt pinii de selecție. Am inițializat modulul LoRa la frecvența utilizată de 868 MHz. Dacă inițializarea eșuează, programul afișează un mesaj de eroare și se blochează.

```
  // Inițializare LoRa
  LoRa.setPins(NSS, -1, DIO0);  // Setează pinii pentru LoRa
  if (!LoRa.begin(868E6)) {  // Setează frecvența de 868 MHz
    Serial.println("Starting LoRa failed!");
    while (1);  // Blochează programul în cazul unei erori
  }
  Serial.println("LoRa inițializat!");
}
```

Figura 36 Inițializare LoRa recepție

În final, am implementat bucla codului care verifică dacă există un pachet de date primit de la modulul LoRa folosind funcția 'LoRa.parsePacket()'. Dacă un pachet este primit, datele sunt citite caracter cu caracter și stocate într-un string. După ce toate datele au fost primite, acestea sunt afișate pe portul serial pentru a putea fi vizualizate și utilizate ulterior.

```
20  void loop() {
21    // Verifică dacă există un pachet primit
22    int packetSize = LoRa.parsePacket();
23    if (packetSize) {
24      // Pachet primit
25      String distance = "";
26      while (LoRa.available()) {
27        distance += (char)LoRa.read();  // Citește datele primite
28      }
29      Serial.print("Distanța recepționată: ");
30      Serial.println(distance);
31
32      // Trimite datele la portul serial
33      Serial.println(distance);
34    }
35  }
```

Figura 37 Recepția și afișarea datelor

5.3.2 Python

Am folosit Python pentru a prelua datele recepționate de la Arduino și a le stoca într-o bază de date InfluxDB pentru analiza și vizualizarea ulterioară, datorită simplității și eficienței sale în manipularea datelor și comunicarea cu baze de date.

Am scris acest script folosind Python 3.12 în Visual Studio, un mediu de dezvoltare integrat.

Pentru a asigura colectarea și stocarea eficientă a datelor, am importat două librării esențiale pentru funcționarea corectă a programului. Am folosit librăria "serial" pentru a

permite comunicarea serială cu Arduino, permițând citirea datelor de la senzorul ultrasonic. Librăria „requests” este utilizată pentru a efectua cereri HTTP, necesare pentru trimiterea datelor către InfluxDB.

```
1   import serial  # Librărie pentru comunicarea serială
2   import requests  # Librărie pentru cereri HTTP
```

Figura 38 Import librării necesare Python

În secțiunea de cod din Figura 39, am configurat parametrii necesari pentru conectarea și trimiterea datelor către baza de date InfluxDB:

- “influxdb_server”: URL-ul serverului InfluxDB la care se conectează.
- „influxdb_token”: Tokenul de autorizare pentru accesul la InfluxDB.
- „influxdb_org”: Numele organizației în cadrul căreia se vor stoca datele.
- „influxdb_bucket”: Numele bucket-ului în care se vor stoca datele.

```
4  # Configurația InfluxDB
5  influxdb_server = "http://192.168.100.11:8086"  # URL-ul serverului InfluxDB
6  influxdb_token = "byZ34CoLu3uO4y8KvOK_0D394sBSC2GRYXD4x3-EkUQnAoc6tsMjXDavoyxh3ZGI9uAEME4WaUg61LCenCZcCQ=="  # Tokenul de autorizare pentru InfluxDB
7  influxdb_org = "proiect_licenta"  # Numele organizației din InfluxDB
8  influxdb_bucket = "date_senzor"  # Numele bucket-ului din InfluxDB
```

Figura 39 Configurare parametrii InfluxDB

În continuare, am configurat portul serial pentru a putea comunica cu Arduino. Am ales portul serial ‘’COM7”, acesta fiind portul la care Arduino este conectat pe laptop-ul meu. De asemenea, am setat o rată de baud de 9600, aceasta trebuie să fie setată la aceeași valoare și în Arduino IDE. Linia de cod „ser=serial.Serial(serial_port, baud_rate)” inițializează conexiunea serială permițând programului să comunice eficient cu Arduino.

```
10  # Configurația portului serial
11  serial_port = 'COM7'  # Portul serial la care este conectat Arduino
12  baud_rate = 9600  # Baud rate-ul pentru comunicația serială
13
14  # Initializează conexiunea serială
15  ser = serial.Serial(serial_port, baud_rate)
```

Figura 40 Configurație port serial Python

În continuare, am definit funcția “send_to_influxdb(value)”, care trimite datele senzorului ultrasonic către InfluxDB. URL-ul serverului și token-ul de autorizare sunt configurate pentru a permite accesul securizat. Funcția utilizează cereri HTTP POST pentru a trimite datele. Codul 204 este utilizat pentru a verifica dacă cererea a fost procesată cu success

de serverul InfluxDB, indicând că datele au fost scrise cu success în baza de date. Dacă scrierea eșuează, se afișează un mesaj de eroare.

```python
def send_to_influxdb(value):
    #Funcție pentru trimiterea datelor către InfluxDB
    url = f"{influxdb_server}/api/v2/write?org={influxdb_org}&bucket={influxdb_bucket}&precision=s"
    headers = {
        "Authorization": f"Token {influxdb_token}",
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = f"ultrasonic_sensor,location=lab value={value}"
    response = requests.post(url, headers=headers, data=data)  # Trimite datele către InfluxDB
    if response.status_code == 204:
        print("Data successfully written to InfluxDB")  # Confirmă scrierea datelor în InfluxDB
    else:
        print(f"Failed to write data to InfluxDB: {response.text}")  # Afișează mesaj de eroare dacă scrierea a eșuat
```

Figura 41 Trimiterea datelor către InfluxDB

În final, am implementat un loop infinit ''while True" pentru a citi în mod continuu date de la portul serial. Fiecare linie citită este decodificată și verificată pentru a vedea dacă conține doar cifre. Dacă datele sunt valide, acestea sunt trimise către InfluxDB folosind funcția "send_to_influxdb(line)". În caz contrar, se afișează un mesaj de eroare specific.

```python
while True:
    try:
        line = ser.readline().decode('utf-8', errors='ignore').strip()  # Citește o linie de pe portul serial
        print(f"Received from serial: {line}")
        if line.isdigit():  # Verifică dacă linia citită conține doar cifre
            send_to_influxdb(line)  # Trimite datele la InfluxDB
        else:
            print(f"Invalid data received: {line}")  # Afișează un mesaj dacă datele primite sunt invalide
    except UnicodeDecodeError as e:
        print(f"Decode error: {e}")  # Afișează mesajul de eroare în cazul unei erori de decodare
    except Exception as e:
        print(f"An error occurred: {e}")  # Afișează orice altă eroare apărută
```

Figura 42 Gestionarea și trimiterea datelor

5.3.3 InfluxDB

Pentru a utiliza baza de date InfluxDB în acest proiect, am început prin pornirea serverului InfluxDB din PowerShell cu comanda "influxd". Aceasta deschide serverul local la adresa "http://localhost:8086", unde mă pot conecta pentru a gestiona și vizualiza datele. Conectarea la localhost înseamnă că serverul rulează pe calculatorul meu, asigurând o comunicare rapidă cu baza de date.

InfluxDB este o bază de date specializată pentru stocarea și interogarea datelor de tip serie temporală, ceea ce o face ideală pentru gestionarea datelor de la senzori în timp real.

În Figura 43, se poate observa cum am creat un query pentru a selecta datele înregistrate de senzorul ultrasonic. Acest query filtrează datele pe baza măsurătorii și a câmpului specificat.

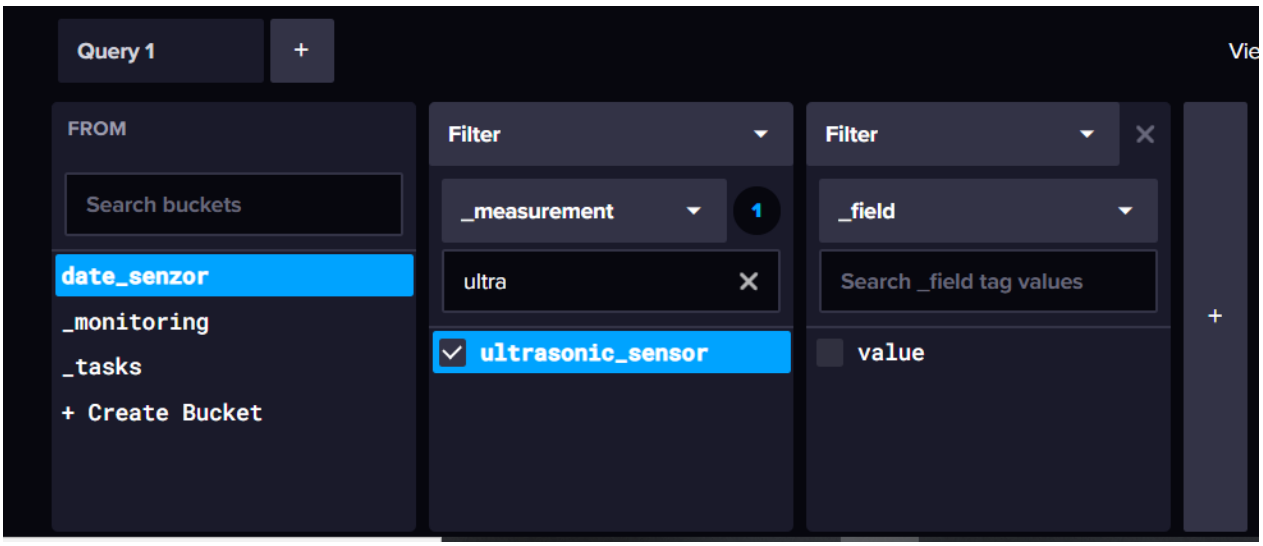

Figura 43 Filtrarea datelor în InfluxDB

5.4 DIAGRAME DE FUNCȚIONARE

5.4.1 Schema generală

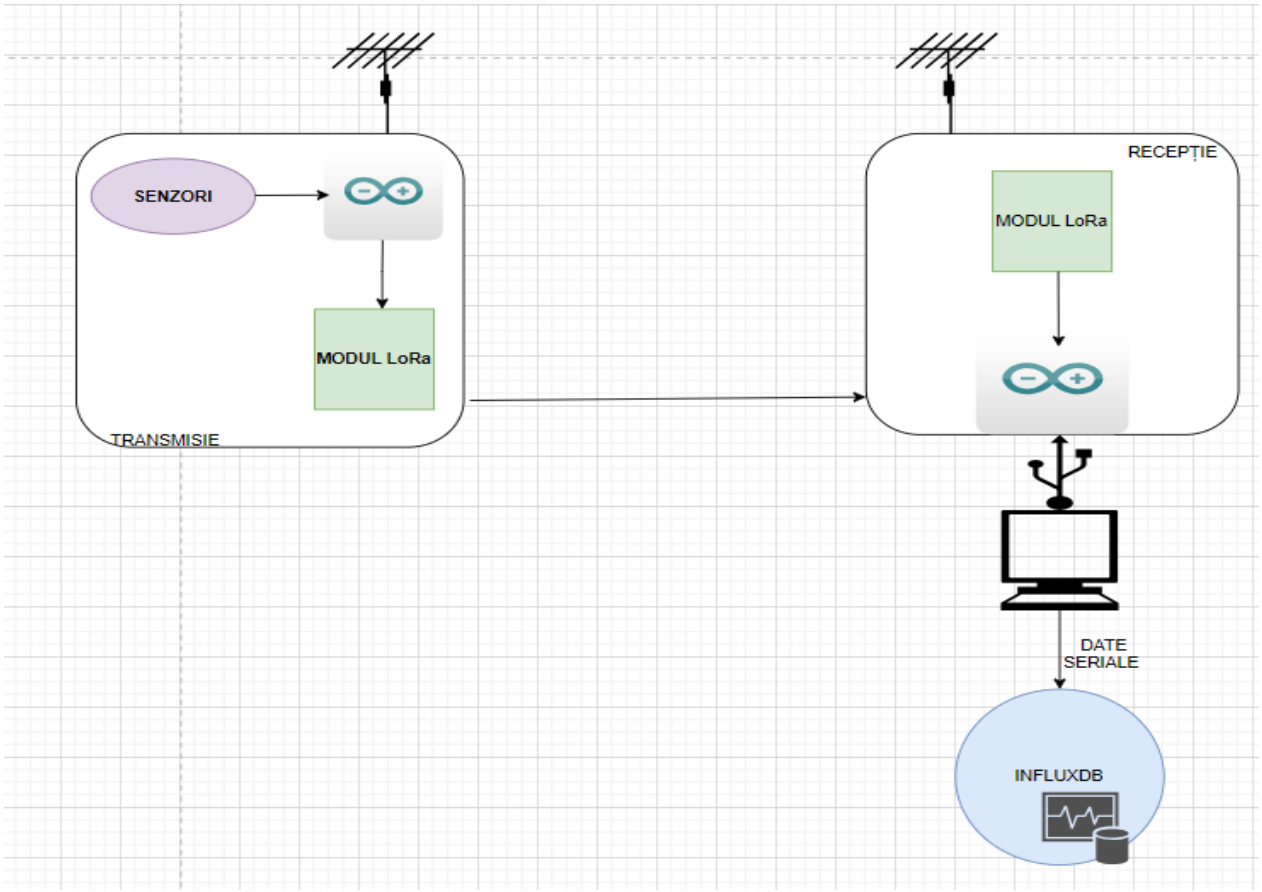

Figura 44 Diagrama de principiu a sistemului

În diagrama de principiu ilustrată în Figura 44, sunt prezentate componentele principale și fluxul de date al proiectului. Sistemul include două părți principale: transmisie și recepție.

1. Transmisie:
   - Senzorii sunt conectați la placa Arduino, care colectează datele de la aceștia.
   - Datele colectate de Arduino sunt trimise către modulul LoRa pentru a fi transmise wireless.
2. Recepție:
   - Modulul LoRa de recepție primește datele transmise de la modulul de transmisie pe o frecvență de 868 MHz.
   - Datele recepționate sunt trimise către placa Arduino, care le trimite mai departe către computer printr-o conexiune USB.
   - Computerul primește datele și le stochează în baza de date InfluxDB pentru monitorizarea și analiza ulterioară.

5.4.2 Diagrama flow – Transmisie

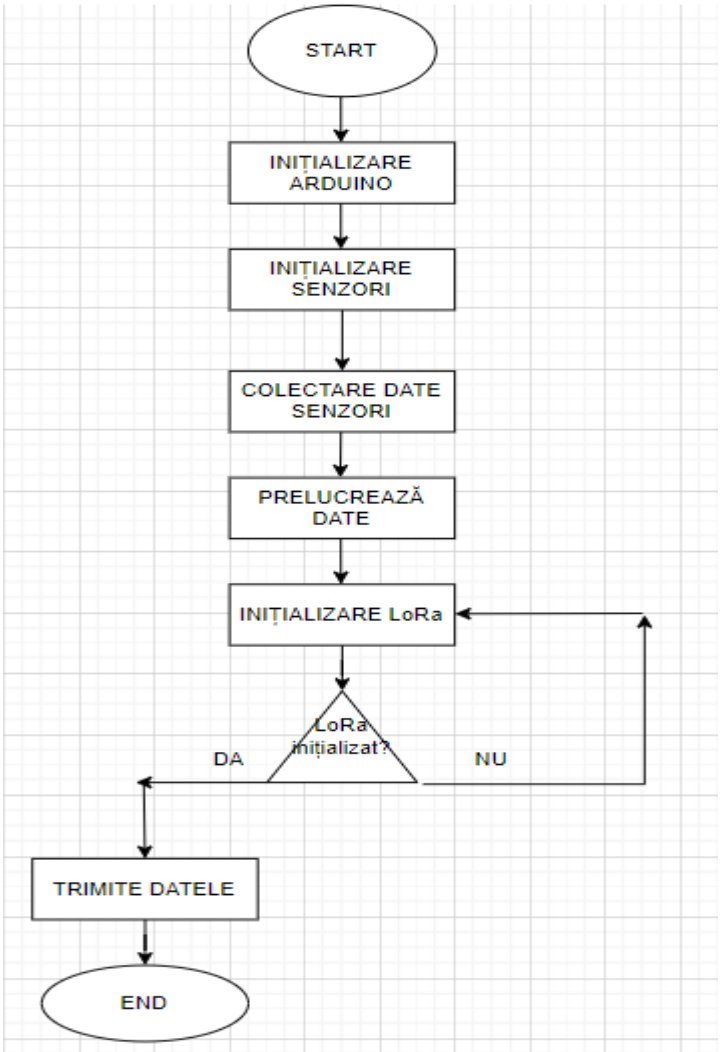

Figura 45 Diagrama flow a transmisiei

Diagrama din Figura 45 reprezintă fluxul de funcționare al sistemului de transmisie a datelor. Aceasta începe cu inițializarea Arduino, urmată de inițializarea senzorilor. După

colectarea și prelucrarea datelor de la senzori, sistemul inițializează modulul LoRa. Dacă inițializarea LoRa este reușită, datele sunt transmise mai departe. În cazul în care inițializarea LoRa eșuează, sistemul revine la etapa de inițializare LoRa, continunând acest proces până la reușită.

5.4.3 Diagrama flow – Recepție

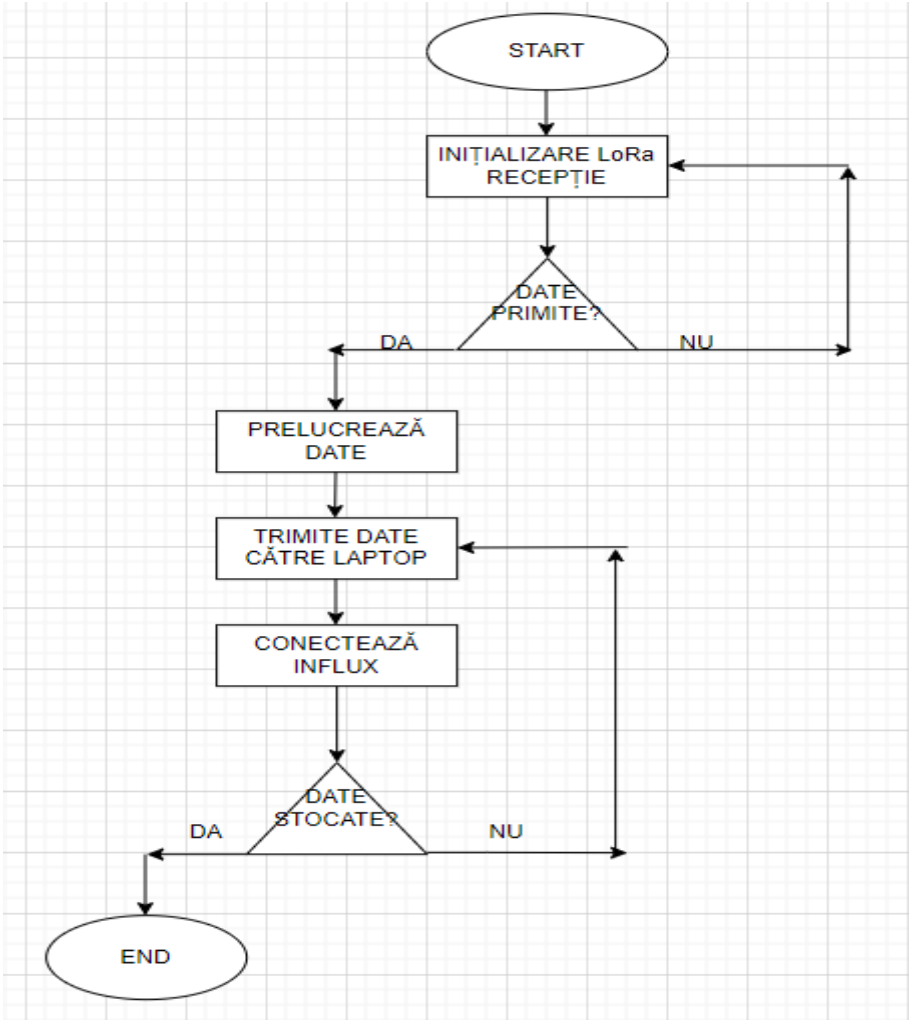

Figura 46 Diagrama flow a recepției

În Figura 46 ilustrează diagrama de flux a procesului de recepție și stocare a datelor în cadrul sistemului. Procesul începe cu inițializarea modului LoRa de recepție. Ulterior, sistemul verifică dacă există date primite. Dacă nu sunt primite date, procesul de reia de la inițializarea LoRa. În cazul în care datele sunt primite, acestea sunt prelucrate și trimise către laptop prin intermediul portului serial. După aceasta, sistemul încearcă să se conecteze la baza de date InfluxDB. Dacă conexiunea este realizată cu succes, datele sunt stocate în baza de date. În caz contrar, procesul se întoarce la trimiterea datelor către laptop pentru o nouă încercare de stocare.

5.4.4 Diagrama flow – Procesare și prelucrare date

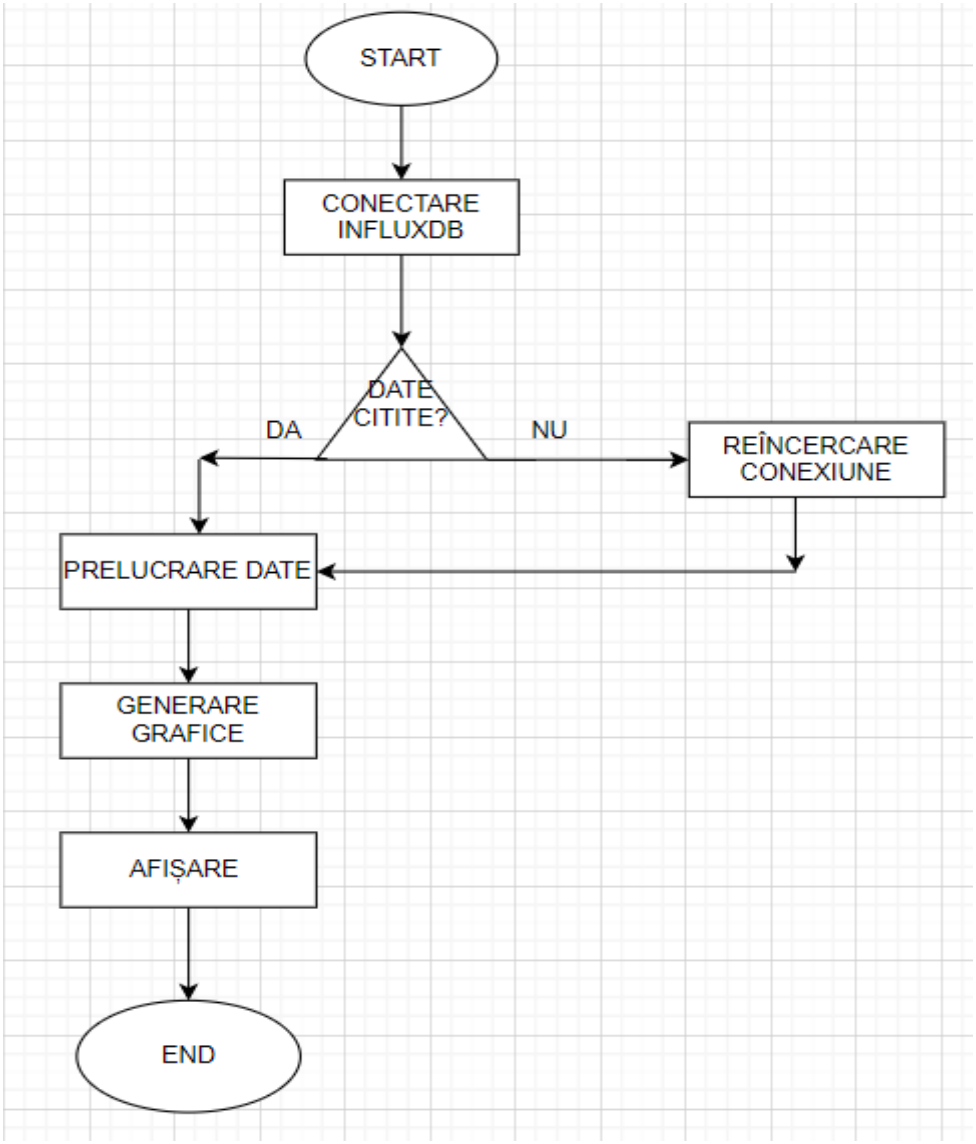

Figura 47 Diagrama flow procesare și prelucrare date

Diagrama din Figura 47 reprezintă fluxul de procesare a datelor stocate în InfluxDB. Începe cu conectarea la baza de date InfluxDB și verifică dacă datele au fost citite corect. Dacă nu, se inițează o reîncercare a conexiunii. În cazul în care datele sunt citite cu succes, acestea sunt prelucrate, urmat de generarea graficelor necesare și afișarea rezultatelor. Procesul de încheie după afișarea graficelor.

6 REZULTATE EXPERIMENTALE

Realizare la nivel hardware
Rezultate la nivel software

6.1 REALIZARE LA NIVEL HARDWARE

În cadrul acestui proiect, am reușit să dezvolt un sistem eficient pentru cartografierea spațiilor inaccesibile, utilizând tehnologia LoRa și senzori ultrasonici.

Partea de transmisie include o plăcuță Arduino conectată la senzori ultrasonici și un modul LoRa, operând la frecvența de 868 MHz. Senzorii colectează date despre distanțe, iar Arduino prelucrează și transmite aceste date prin modulul LoRa.

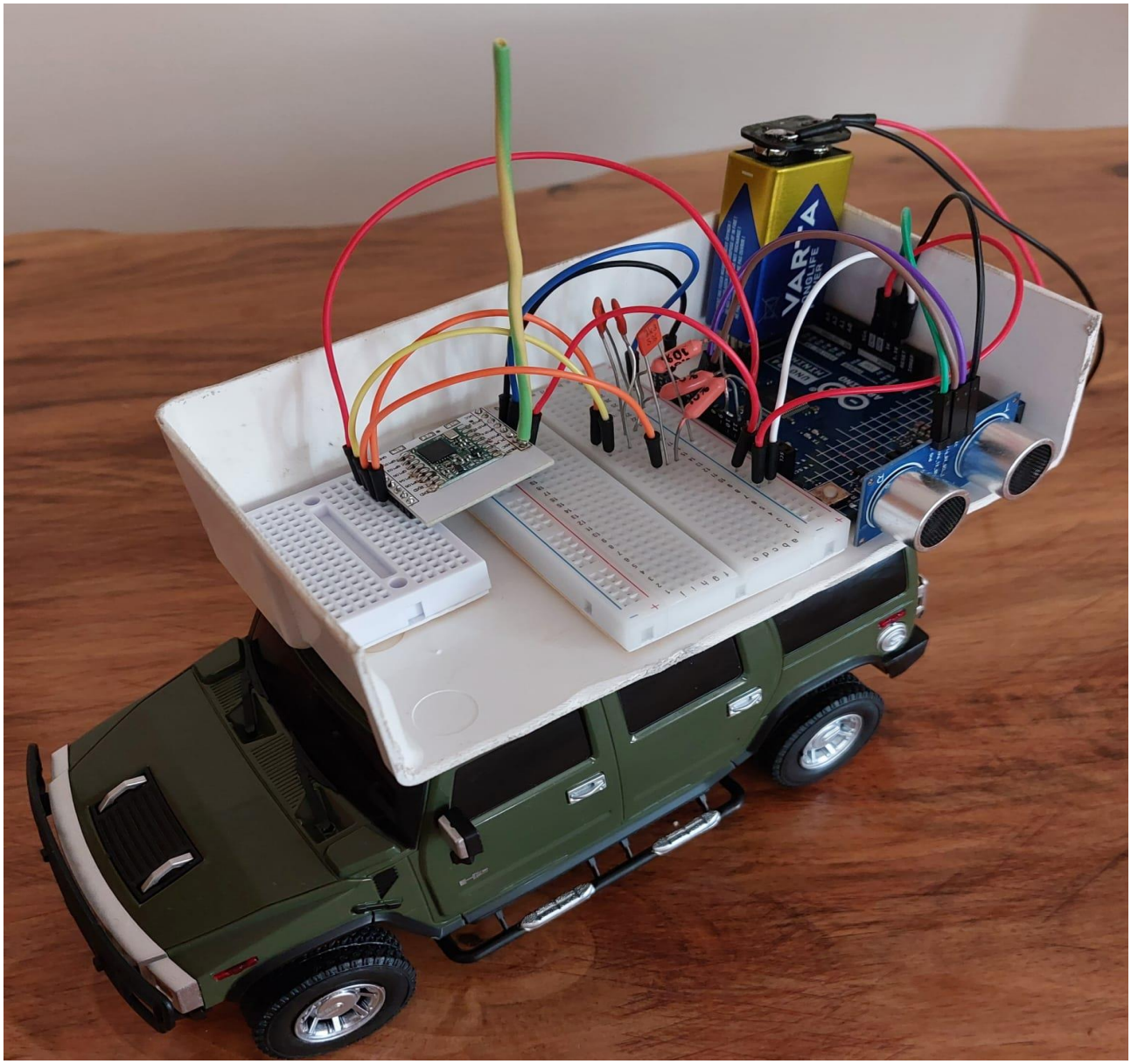

Figura 48 Partea de transmisie realizată

Partea de recepție constă într-o altă plăcuță Arduino conectată la un modul LoRa și un laptop. Modulul LoRa primește datele transmise la 868 MHz, Arduino le prelucrează și le trimite către laptop pentru stocare și analiză în InfluxDB.

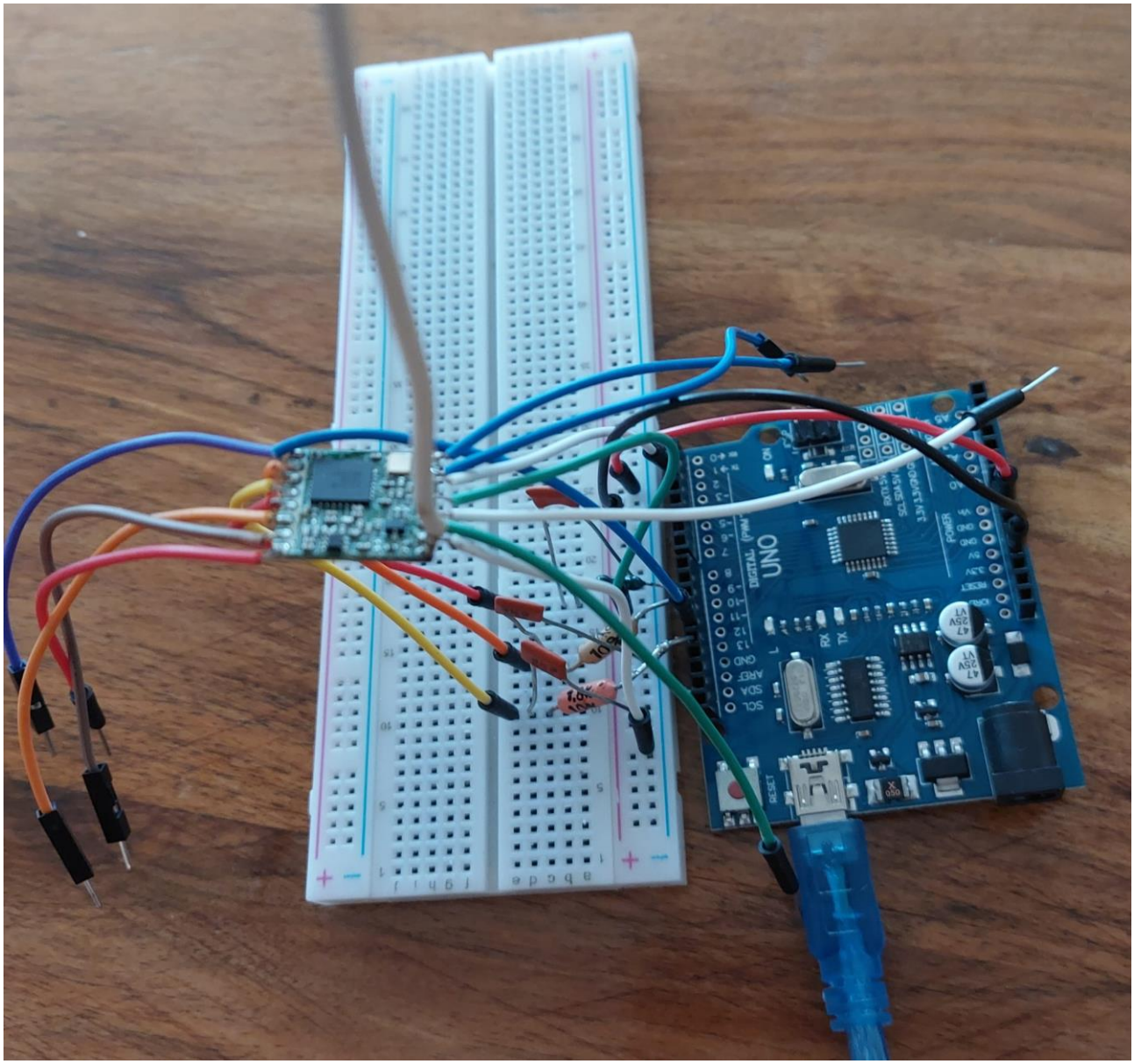

Figura 49 Partea de recepție realizată

6.2 REZULTATE LA NIVEL SOFTWARE

În acest subcapitol, voi prezenta rezultatele obținute la nivel software, împărțite în utilizarea Arduino IDE și analiza datelor în InfluxDB.

6.2.1 Arduino IDE

În cadrul Arduino IDE, am monitorizat datele primate de la senzori și transmise prin modulul LoRa. Aceste date sunt afișate în serial monitor, oferind o vedere în timp real a datelor colectate.

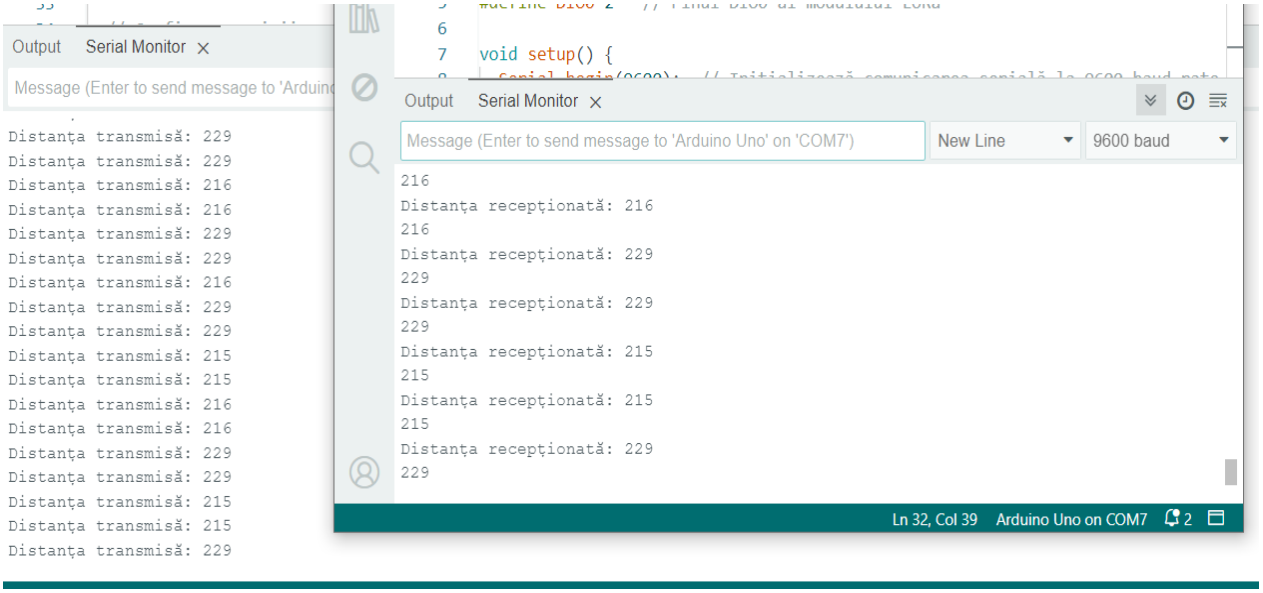

Figura 50 Rezultate Arduino IDE

6.2.2 InfluxDB

Datele recepționate și prelucrate sunt apoi stocate și analizate folosind InfluxDB.

În Figura 51 este prezentat graficul care ilustrează modul în care datele colectate sunt vizualizate, permițând o analiză în timp real a distanțelor măsurate de senzori.

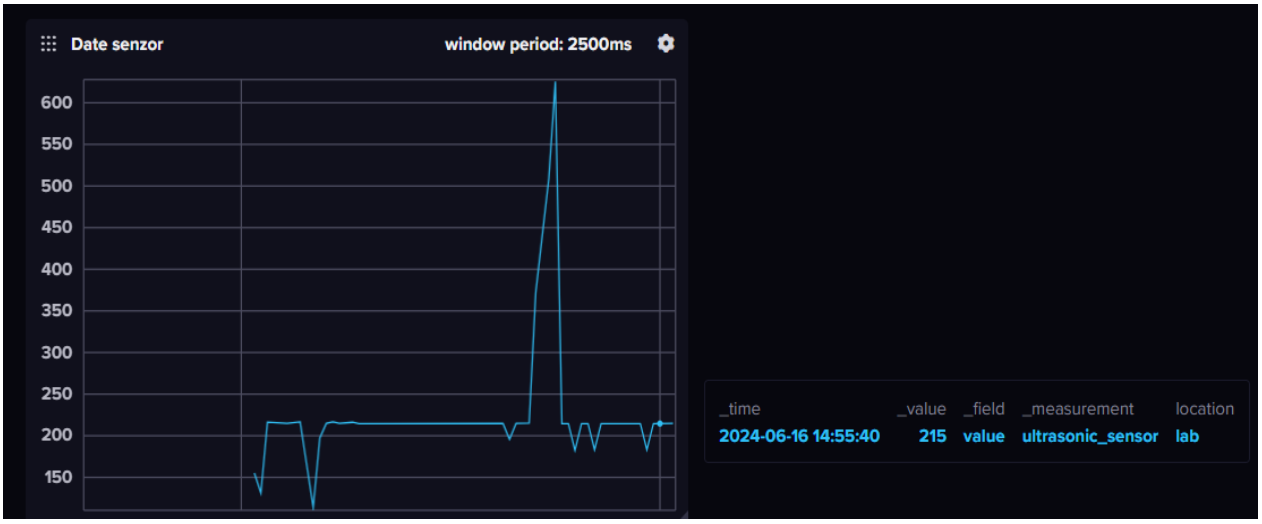

Figura 51 Rezultate grafice

7 CONCLUZII

Posibile dezvoltări
Raport tehnico-economic

În cadrul acestui proiect, am reușit să dezvolt un sistem eficient pentru cartografierea spațiilor inaccesibile, folosind tehnologia de comunicație LoRa. Alegerea acestei tehnologii s-a bazat pe necesitatea de a transmite date pe distanțe mari, cu un consum redus de energie, ceea ce o face ideală pentru utilizarea în medii izolate, cum ar fi peșterile. Sistemul include două părți esențiale: una pentru transmisie, care colectează și transmite datele senzorilor cu ultrasunete, și una pentru recepție, care primește și procesează aceste date.

7.1 POSIBILE DEZVOLTĂRI

Pentru a îmbunătăți și mai mult sistemul, se poate evolua în câteva direcții:

- Adăugarea mai multor senzori cu ultrasunete care ar permite realizarea unei cartografieri tridimensionale a spațiilor.
- Integrarea altor tipuri de senzori, cum ar fi de temperatură, umiditate sau gaze, astfel făcând sistemul mai util în monitorizarea mediului sau a condițiilor de siguranță din peșteri.
- Implementarea unei interfețe web sau a unei aplicații mobile pentru monitorizarea în timp real al sistemului.
- Dezvoltarea unui mecanism de auto-salvare pentru mașinuță, care să îi permit să se întoarcă în cazul în care rămâne blocată sau să transmită un semnal de avertizare către operator.

7.2 RAPORT TEHNICO-ECONOMIC

Analizând din punct de vedere tehnic și economic, sistemul propus oferă o soluție rentabilă pentru cartografierea spațiilor inaccesibile.

În Tabelul 6 se pot observa costurile realizării acestui sistem.

Tabelul 6 Costurile sistemului

Componentă	Bucăți	Preț per bucată (RON)	Preț total per bucăți(RON)
Arduino UNO R4 Minima	1	105	105
Arduino UNO R3	1	53	53
Modul RFM95	2	33	66
Senzor HC-SR04	1	10	10
Rezistori	12	0,10	1,2
Mașină telecomandă	1	100	100
Breadboard	3	4	12
Fire mamă – tată	1	12	12
Placa adaptoare ESP8266	1	2.5	2.5
Cablu USB A-B	1	8	8
Baterie 9V	1	11	11
Soclu baterie	1	2	2
PREȚ TOTAL SISTEM		383 RON	

8 BIBLIOGRAFIE

[1] Despre comunicațiile fără fir - https://ro.scribd.com/document/369299979/Comunicatii-Fara-Fir

[2] Informații LoRaWAN - https://www.semtech.com/uploads/technology/LoRa/lora-and-lorawan.pdf

[3] *"LoRa Modulation Basics"* (PDF). *Semtech*. *Archived from the original* (PDF) on 2019-07-18. *Retrieved 2020-02-05.*

[4] Mihai Romanca - *Microprocesoare și microcontrolere*, Editura Universității Transilvania, Brașov, 2015.

[5] Datasheet RA4M1 - https://pdf1.alldatasheet.com/datasheet-pdf/view/1275915/RENESAS/RA4M1.html

[6] Datasheet Atmega328P - https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[7] Bârlea, N.-M., Fizica Senzorilor, Ed. Albastră, Cluj-Napoca 2000, ISBN 973-9443-427

[8] https://electricalc.ro/panouri-fotovoltaice/23-blog/63-bazele-masurarii-aparate-de-masura-caracteristici-metrologice

[9] Datasheet Arduino UNO R4 Minima- https://docs.arduino.cc/resources/datasheets/ABX00080-datasheet.pdf

[10] Datasheet Arduino UNO R3 - https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[11] Datasheet RFM95 - https://pdf1.alldatasheet.com/datasheet-pdf/view/1132605/HOPE/RFM95.html

[12] Morgan, E. J. (2014). HC-SR04 Ultrasonic Sensor - https://pdf1.alldatasheet.com/datasheet-pdf/view/1132204/ETC2/HCSR04.html

[13] Imagine senzori - https://www.hwlibre.com/ro/hc-sr04/

[14] Divizorul de tensiune - https://sites.google.com/site/bazeleelectronicii/home/circuite-dc-si-teoria-circuitelor-dc/15-divizoare-de-tensiune

[15] Imagine divizor de tensiune - https://www.google.ro/search?sca_esv=0b05e7fa75492358&hl=ro&sxsrf=ADLYWIKHiUbAbvM2B1eWOGhKBk0qeQnQbw:1718565925395&q=divizorul+de+tensiune&udm=2&fbs=AEQNm0AuaLfhdrtx2b9ODfK0pnmis1zS4enB7jefi_fubH5nzzN3wC83IX-0M5KItd9aV1roE9loLEqXbKR-DA_2Bgy7U7IEDeDOIYsA7etVtunHFyxwIt9noukItdAzPmjdQ21aU68IgzQkeRdeOEnvWGTow8KSzQ-

JOvQ8eUkbDTzSRs-eDchqQ_Bc9chLwJKXRw0EF2SZ&sa=X&ved=2ahUKEwjOjIb87OCGAxU1_rsIHbpBCOoQtKgLegQIDBAB&biw=1536&bih=703&dpr=1.25#vhid=IddMBlzwxUT1ZM&vssid=mosaic

[16] Arduino IDE - https://ro.scribd.com/document/551329672/Arduino-IDE

[17] Python - https://www.w3schools.com/python/python_intro.asp

[18] InfluxDB - https://awesome.influxdata.com/docs/part-1/introduction-to-influxdb/#what-is-influxdb

[19] Creare nod LoRa - https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_build_lora_node_rfm95_arduino_uno.html

9 ANEXE

9.1 COD SURSĂ ARDUINO

9.1.1 Transmisie

```
#include <SPI.h>  // Librărie pentru comunicarea SPI
#include <LoRa.h> // Librărie pentru comunicarea LoRa

#define NSS 10   // Pinul de selectare a modulului LoRa
#define DIO0 2   // Pinul DIO0 al modulului LoRa
#define MOSI_PIN 11 // Pinul MOSI (Master Out Slave In) pentru SPI
#define MISO_PIN 12 // Pinul MISO (Master In Slave Out) pentru SPI
#define SCK_PIN 13  // Pinul SCK (Serial Clock) pentru SPI

#define TRIG_PIN 6  // Pinul TRIG al senzorului ultrasonic
#define ECHO_PIN 7  // Pinul ECHO al senzorului ultrasonic

void setup() {
  Serial.begin(9600);  // Inițializează comunicarea serială la 9600 baud rate
  while (!Serial);  // Așteaptă deschiderea conexiunii seriale

  Serial.println("Inițializare LoRa...");

  // Configurare pinii ca ieșire
  pinMode(NSS, OUTPUT);
  digitalWrite(NSS, HIGH);

  pinMode(DIO0, INPUT);

  // Configurare pini SPI
  pinMode(MOSI_PIN, OUTPUT);
  digitalWrite(MOSI_PIN, HIGH);

  pinMode(SCK_PIN, OUTPUT);
  digitalWrite(SCK_PIN, HIGH);

  pinMode(MISO_PIN, INPUT);

  // Configurare pinii senzorului ultrasonic
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Inițializare SPI
  SPI.begin();

  // Inițializare LoRa
  if (!LoRa.begin(868E6)) {  // Setează frecvența de 868 MHz
    Serial.println("Inițializarea LoRa a eșuat!");
    while (1);  // Blochează programul în cazul unei erori
  }
  Serial.println("LoRa inițializat!");
}

void loop() {
  long duration, distance;

  // Trimitere puls TRIG
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Citire durata pulsului ECHO
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1;  // Calculare distanță în cm

  // Trimitere mesaj LoRa
  Serial.print("Distanța transmisă: ");
  Serial.println(distance);
  LoRa.beginPacket();
  LoRa.print(distance);
  LoRa.endPacket();
```

```
  delay(1000); // Delay de 1 secundă
}
```

Codul 1. Cod sursă Arduino pentru transmisie

9.1.2 Recepție

```
#include <SPI.h>  // Librărie pentru comunicarea SPI
#include <LoRa.h> // Librărie pentru comunicarea LoRa

#define NSS 10   // Pinul de selectare a modulului LoRa
#define DIO0 2   // Pinul DIO0 al modulului LoRa

void setup() {
  Serial.begin(9600);  // Inițializează comunicarea serială la 9600 baud rate
  while (!Serial);  // Așteaptă deschiderea conexiunii seriale

  // Inițializare LoRa
  LoRa.setPins(NSS, -1, DIO0);  // Setează pinii pentru LoRa
  if (!LoRa.begin(868E6)) {  // Setează frecvența de 868 MHz
    Serial.println("Starting LoRa failed!");
    while (1);  // Blochează programul în cazul unei erori
  }
  Serial.println("LoRa inițializat!");
}

void loop() {
  // Verifică dacă există un pachet primit
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // Pachet primit
    String distance = "";
    while (LoRa.available()) {
      distance += (char)LoRa.read();  // Citește datele primite
    }
    Serial.print("Distanța recepționată: ");
    Serial.println(distance);

    // Trimite datele la portul serial
    Serial.println(distance);
  }
}
```

Codul 2. Codul sursă Arduino pentru recepție

9.2 COD SURSĂ PYTHON

```
import serial  # Librărie pentru comunicarea serială
import requests  # Librărie pentru cereri HTTP

# Configurația InfluxDB
influxdb_server = "http://192.168.100.11:8086"  # URL-ul serverului InfluxDB
influxdb_token = "byZ34CoLu3uO4y8KvOK_0D394sBSC2GRYXD4x3-
EkUQnAoc6tsMjXDavoyxh3ZGI9uAEME4WaUg61LCenCZcCQ=="  # Tokenul de autorizare pentru InfluxDB
influxdb_org = "proiect_licenta"  # Numele organizației din InfluxDB
influxdb_bucket = "date_senzor"  # Numele bucket-ului din InfluxDB

# Configurația portului serial
serial_port = 'COM7'  # Portul serial la care este conectat Arduino
baud_rate = 9600  # Baud rate-ul pentru comunicația serială

# Initializează conexiunea serială
ser = serial.Serial(serial_port, baud_rate)

def send_to_influxdb(value):
    #Funcție pentru trimiterea datelor către InfluxDB
    url = f"{influxdb_server}/api/v2/write?org={influxdb_org}&bucket={influxdb_bucket}&precision=s"
    headers = {
        "Authorization": f"Token {influxdb_token}",
        "Content-Type": "application/x-www-form-urlencoded"
    }
    data = f"ultrasonic_sensor,location=lab value={value}"
    response = requests.post(url, headers=headers, data=data)  # Trimite datele către InfluxDB
    if response.status_code == 204:
        print("Data successfully written to InfluxDB")  # Confirmă scrierea datelor în InfluxDB
```

```
    else:
        print(f"Failed to write data to InfluxDB: {response.text}")  # Afișează mesaj de eroare dacă
scrierea a eșuat

while True:
    try:
        line = ser.readline().decode('utf-8', errors='ignore').strip()  # Citește o linie de pe
portul serial
        print(f"Received from serial: {line}")
        if line.isdigit():  # Verifică dacă linia citită conține doar cifre
            send_to_influxdb(line)  # Trimite datele la InfluxDB
        else:
            print(f"Invalid data received: {line}")  # Afișează un mesaj dacă datele primite sunt
invalide
    except UnicodeDecodeError as e:
        print(f"Decode error: {e}")  # Afișează mesajul de eroare în cazul unei erori de decodare
    except Exception as e:
        print(f"An error occurred: {e}")  # Afișează orice altă eroare apărută
```

Codul 3. Codul sursă Python

REZUMAT

Proiectul intitulat „Sistem de cartografiere a spațiilor inaccesibile și transmisie prin LoRa" își propune să dezvolte un vehicul autonom echipat cu senzori și module de comunicație LoRa pentru explorarea și cartografierea unei peșteri. Acest sistem integrează hardware și software pentru a realiza măsurători precise și a transmite datele colectate în timp real. Vehiculul este echipat cu senzori ultrasonici HC-SR04 conectați la un microcontroler Arduino UNO, care gestionează achiziția și transmiterea datelor prin module LoRa RFM95.

Datele sunt transmise către un al doilea modul LoRa conectat la un laptop, unde sunt stocate într-o bază de date InfluxDB pentru vizualizare și analiză. Utilizarea tehnologiei LoRa permite transmiterea datelor pe distanțe mari cu un consum redus de energie, ceea ce face sistemul ideal pentru medii dificile. Comunicarea LoRa la frecvența de 868 MHz asigură o transmisie bună a semnalului prin obstacole, fiind adecvată pentru mediile subterane ale peșterilor.

Partea de software a proiectului include programarea microcontrolerelor în Arduino IDE pentru gestionarea senzorilor și a modulelor de comunicație, precum și utilizarea unui script Python pentru prelucrarea datelor și transmiterea acestora către InfluxDB. Vizualizarea datelor în InfluxDB permite analiza ușoară și rapidă a măsurătorilor realizate.

Posibile dezvoltări viitoare includ integrarea mai multor senzori pentru a permite o cartografiere tridimensională detaliată a spațiilor explorate. De asemenea, se poate implementa un mecanism de detectare a defecțiunilor și autoredresare a vehiculului în cazul blocării sau a altor probleme tehnice, asigurând astfel o explorare continuă și fiabilă. Proiectul demonstrează cum tehnologiile moderne pot fi aplicate eficient pentru explorarea și documentarea mediilor inaccesibile.

ABSTRACT

The project titled "System for Mapping Inaccessible Spaces and Transmission via LoRa" aims to develop an autonomous vehicle equipped with sensors and LoRa communication modules for exploring and mapping a cave. This system integrates hardware and software to achieve precise measurements and transmit the collected data in real-time. The vehicle is equipped with HC-SR04 ultrasonic sensors connected to an Arduino UNO microcontroller, which manages the data acquisition and transmission through LoRa RFM95 modules.

The data is transmitted to a second LoRa module connected to a laptop, where it is stored in an InfluxDB database for visualization and analysis. The use of LoRa technology allows for long-distance data transmission with low power consumption, making the system ideal for challenging environments. LoRa communication at the 868 MHz frequency ensures good signal transmission through obstacles, making it suitable for the underground environments of caves.

The software part of the project includes programming the microcontrollers in Arduino IDE for managing the sensors and communication modules, as well as using a Python script for processing the data and sending it to InfluxDB. Data visualization in InfluxDB allows for easy and quick analysis of the measurements taken.

Possible future developments include integrating more sensors to allow detailed three-dimensional mapping of the explored spaces. Additionally, a fault detection and self-recovery mechanism can be implemented for the vehicle in case of blockage or other technical problems, thus ensuring continuous and reliable exploration. The project demonstrates how modern technologies can be effectively applied for the exploration and documentation of inaccessible environments.