КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (24), 2024
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2024.24.241256
УДК 004.057.4:004.056

Крючкова Лариса Петрівна
д.т.н., професор, професор кафедри інформаційної та
кібернетичної безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, Київ, Україна
ORCID ID: 0000-0002-8509-6659
l.kriuchkova@kubg.edu.ua

Леонтюк Нікіта Сергійович
студент
Київський столичний університет імені Бориса Грінченка, Київ, Україна
nsleontiuk.fitu20@kubg.edu.ua

ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ШВИДКОЇ ОЦІНКИ ПОТУЖНОСТІ WI-FI СИГНАЛУ В ТОЧКАХ ПРОСТОРУ УРБАНІЗОВАНОГО ПРИМІЩЕННЯ

Анотація. У зв'язку з широким використанням безпроводових технологій Wireless Fidelity виникає актуальна проблема забезпечення належного рівня Wi-Fi сигналу в просторі урбанізованих приміщень. Наявність усередині будівлі стін, перегородок, меблів, радіоелектронної апаратури та інших об'єктів ускладнює умови поширення радіохвиль. Основними ефектами, що спостерігаються при поширенні радіохвиль всередині приміщень, є багатопроменевість, обумовлена багаторазовими відбиттями радіохвиль від стін та інших об'єктів, дифракція на численних гострих краях предметів і згасання радіохвиль при поширенні та при проходженні через перешкоди. Для ефективного використання безпроводових мереж у зазначених умовах необхідно мати можливість швидкої оцінки рівня Wi-Fi сигналу в просторі приміщення. Мета публікації — програмно-апаратна реалізація алгоритму швидкої оцінки потужності Wi-Fi сигналу в множині точок простору урбанізованого приміщення. Представлено варіант роботизованої платформи з необхідною електронікою та програмним забезпеченням, здатної автоматично здійснювати швидку оцінку потужності Wi-Fi сигналу в заданій множині точок урбанізованого простору; блок-схему узагальненого алгоритму роботи робота; програмні коди реалізацій функцій робота та алгоритму оцінки потужності Wi-Fi сигналу. Запропонована методика швидкої оцінки потужності Wi-Fi сигналу в точках простору, відокремлених перешкодами з різними коефіцієнтами згасання, характеризується низькими обчислювальними витратами, що з успіхом може бути використано для оптимізації розташування точок доступу в межах приміщення із заданою геометрією і забезпечення стійкого покриття простору приміщення Wi-Fi-сигналом. Подальший технічний прогрес у створенні більш досконалої електроніки дозволить здійснити модифікацію робота для підвищення точності орієнтування у просторі і точності оцінки потужності Wi-Fi-сигналу, що підвищить ефективність його застосування.

Ключові слова: безпроводові технології; оцінка потужності Wi-Fi сигналу; роботизована платформа; урбанізоване приміщення; множина точок простору; алгоритм; програмний код.

ВСТУП

У зв'язку з широким використанням безпроводових технологій Wireless Fidelity (Wi-Fi), виникає актуальна проблема забезпечення належного рівня Wi-Fi сигналу в просторі урбанізованих приміщень. Для оптимізації розташування точок доступу в межах приміщення із заданою геометрією необхідно мати можливість швидкої та точної оцінки рівня Wi-Fi сигналу в різних зонах приміщення.

ISSN 2663 - 4023

Як відомо, поширення в оточуючому просторі електромагнітної хвилі супроводжується перенесенням енергії. Напрямок (рис. 1) і величина густини потоку потужності електромагнітного поля [Вт/м2] від джерела Q в кожній точці простору визначається вектором Пойнтінга $\vec{\Pi}$, який є результатом векторного добутку векторів напруженості електричного($\vec{\mathbf{E}}$) і магнітного ($\vec{\mathbf{H}}$) полів, тобто утворює разом з ними праву трійку векторів [1]:

$$\vec{\Pi} = \left[\vec{E}, \vec{H}\right]$$

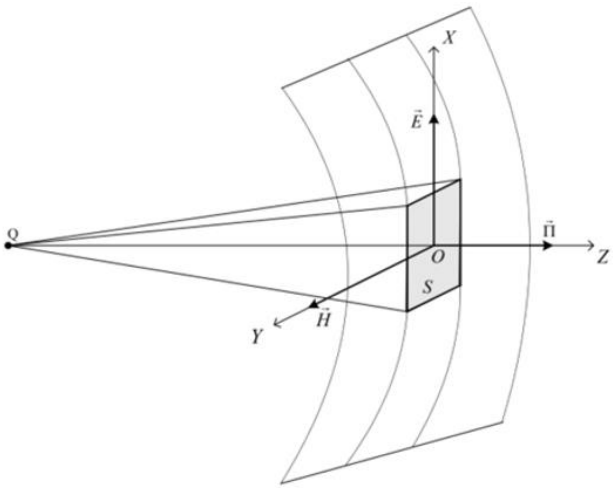

Рис. 1. Вектор Пойнтінга

Наявність усередині будівлі стін, перегородок, меблів, радіоелектронної апаратури та інших об’єктів ускладнює умови поширення радіохвиль. Основними ефектами, що спостерігаються при поширенні радіохвиль всередині урбанізованих приміщень, є багатопроменевість, обумовлена багаторазовими відображеннями радіохвиль від стін та інших об’єктів, дифракція на численних гострих краях предметів і згасання радіохвиль при поширенні та при проходженні через перешкоди. Ці ефекти зумовлюють складну структуру електромагнітного поля [1]. Очевидно, що аналітичний розрахунок рівня Wi-Fi сигналу в просторі урбанізованого приміщення є досить складним.

Існують різноманітні Wi-Fi аналізатори [2] у вигляді програм, окремих пристроїв та додатків для смартфонів і планшетів, здатних сканувати навколишні Wi-Fi мережі і відображати їх параметри, в тому числі й потужність сигналу. Однак вони не забезпечують достатню деталізацію даних та мають просторову обмеженість. Аналізатори спеціального спрямування, призначені для моніторингу радіовипромінювань у приміщеннях, не відображують просторової картини поля.

Мета публікації — програмно-апаратна реалізація алгоритму швидкої оцінки потужності Wi-Fi сигналу в множині точок простору урбанізованого приміщення.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Нами пропонується для вирішення зазначеної задачі використовувати роботизовану платформу (далі — робот) із необхідною для вимірювань електронікою та програмним забезпеченням, здатної забезпечити визначення точок простору, де потужність Wi-Fi сигналу недостатня для забезпечення якісного доступу до мережі Internet. Просторова картина рівня Wi-Fi сигналу дозволить оптимізувати розташування

точок доступу в межах приміщення із заданою геометрією і забезпечити стійке покриття простору приміщення Wi-Fi сигналом.

Для кращої прохідності, із врахуванням перешкод на рівні підлоги (пороги, килими, малі і плоскі предмети), нами вибрано для роботизованої платформи чотириноге крокуюче шасі [3], що вже зарекомендувало себе у багатьох варіантах роботів схожої конструкції. Перевагами крокуючого шасі є висока прохідність, стійкість та маневреність. До недоліків такого рішення слід віднести меншу порівняно з колісним та гусеничним шасі швидкість переміщення у просторі та високу вартість. Тому, якщо умови дозволяють, доцільним буде застосування колісного або гусеничного шасі.

Для виконання задачі із оцінки потужності Wi-Fi сигналу, в якості імітатора Wi-Fi сигналу використовується передавач, який розміщується роботом в заданій точці простору, та відповідний приймач, розташований на роботизованій платформі. Окрім того, роботизована платформа має бути обладнана необхідними для орієнтування у просторі датчиками (далекоміри, GPS-навігація, комп'ютерний зір), мікропроцесором та іншою електронікою, забезпечених достатнім рівнем живлення від акумулятора.

Компонування робота

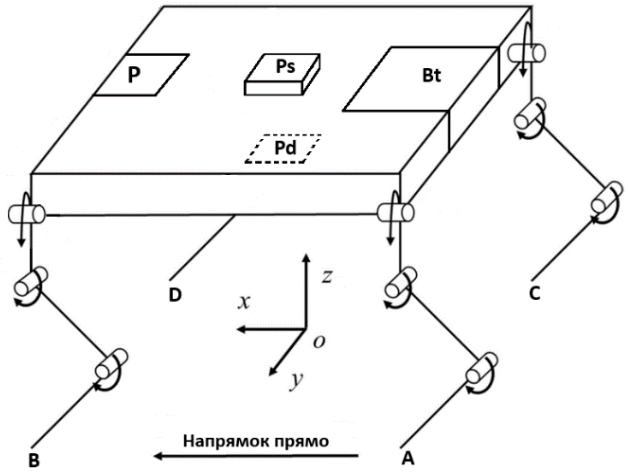

Рис. 2. Схема компонування робота (A, B, C, D — опори опорно-рухової системи; Pd — транспортний відсік із передавачем; P — плата з процесором, панель керування та комплекс датчиків; Ps — башта з приймачем; Bt — блок живлення)

Опорно-рухова система — чотири опори у вигляді кінцівок з'єднані із корпусом робота, що здатні згинатися. Загалом аналогічна до робо-собак серії *Spot* [4] від компанії *BostonDynamics* або подібних.

Транспортний відсік — відсік невеликого розміру в нижній частині корпусу посередині між опорами. Необхідний для транспортування та розміщення передавача.

Передавач — складається з генератора електромагнітних хвиль та невеликої плати для керування його роботи. Знаходиться в транспортному відсіку.

Плата з процесором — з'єднує всю електроніку робота та виконує обчислювальні функції, знаходиться у верхній передній частині корпусу.

Комплекс датчиків — набір датчиків для орієнтування в просторі, що знаходиться у передній та бокових частинах корпусу робота. Складається із ультразвукових датчиків відстані [5].

Башта з приймачем — знаходиться у верхній частині корпусу, посередині та на невеликому підвищенні. Необхідна для прийняття сигналу з передавача.

ISSN 2663 - 4023

Панель керування — знаходиться у верхній передній частині корпусу біля плати з процесором. Складається з LCD-екрану та клавіатури.

Блок живлення — акумулятор або комплекс акумуляторів, що живить собою всю електроніку робота. Розташований у задній частині корпусу робота.

Схематичне зображення макетної плати електронної частини робота подано на рис. 3.

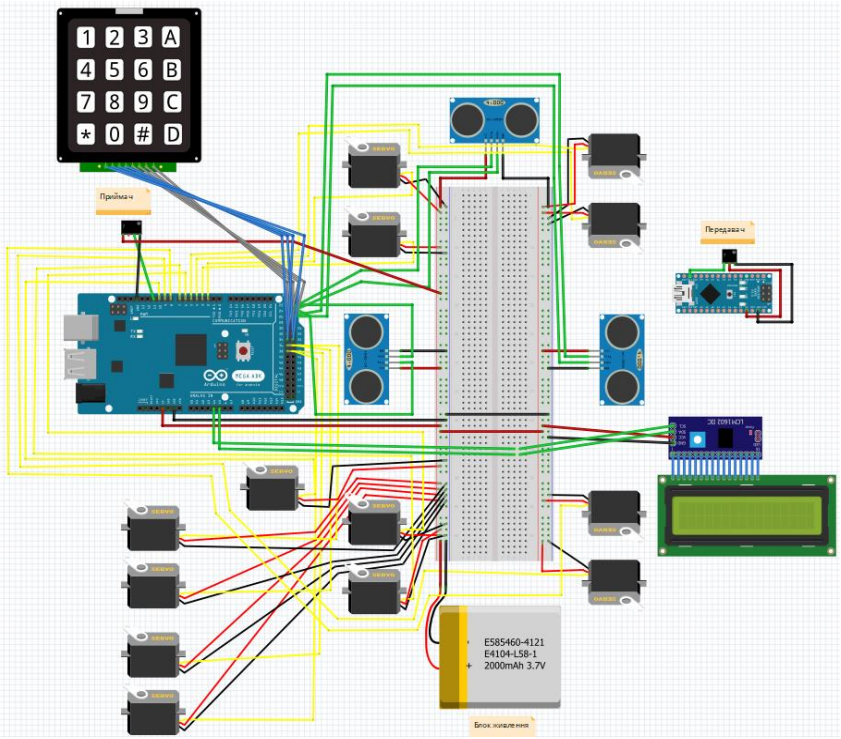

Рис. 3. Схематичне зображення макетної плати електронної частини робота

В якості елементів компонування робота, використовувалась наступна електроніка:

- Arduino MEGA ADK (Rev3) [6] — плата з процесором;
- LIPO-2000mAh — блок живлення;
- HC-CR04 [7] (x3) — ультра-звукові датчики відстані (комплекс датчиків);
- LCD-екран та конвертор в I2C [8] (панель керування);
- Matrix Keypad [9] — клавіатура (панель керування);
- MX-RM-5V [10] — приймач;
- FS1000A — передавач;
- Arduino UNO — плата з процесором для роботи передавача;
- Basic Servo (x13) — сервомотори (опорно-рухова система, нижній транспортний відсік).

Узагальнений алгоритм роботи робота

Крок №1. Початок роботи, введення користувачем координат про місце розміщення передавача.

Крок №2. Старт руху робота до точки розміщення передавача в якості майбутнього місцезнаходження Wi-Fi-роутера.

Крок №3. Введення користувачем координат множини точок простору.

Крок №4. Переміщення по координатах точок простору.

Крок №5. Паралельне виявлення перешкод, в разі зустрічі виконується оминання та корегується маршрут.

Крок №6. По досягненню точки простору відбувається прийняття сигналу з передавача. Якщо маршрут не пройдено, то повторювати крок №4.

Крок №5. По закінченню маршруту робот повертається до точки розміщення передавача та забирає його до відсіка.

Крок №6. Оцінювання сигналу в пройдених точках простору, згідно отриманих даних. Виведення результатів.

Блок-схема узагальненого алгоритму роботи робота подана на рис. 4.

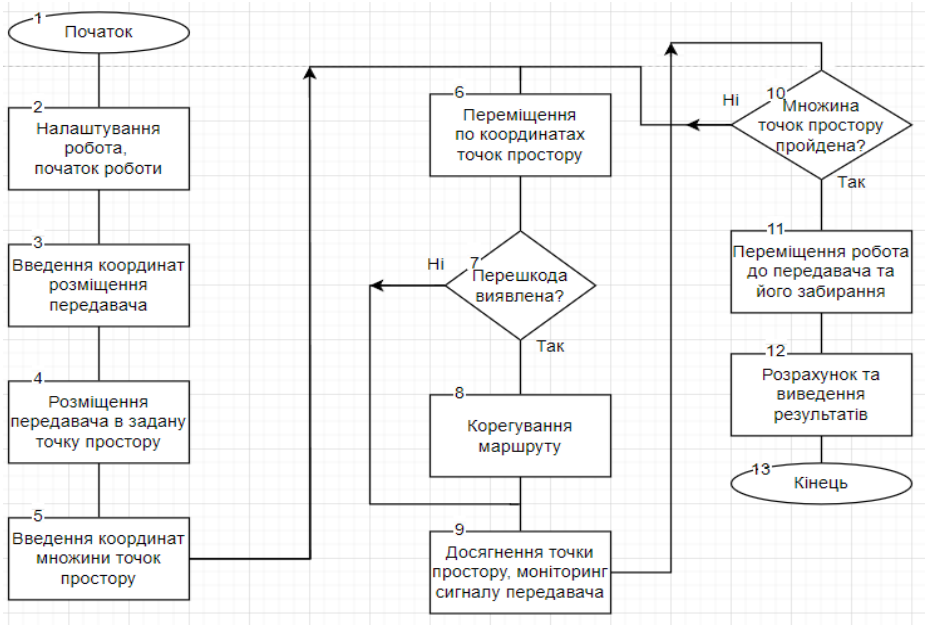

Рис. 4. Блок-схема узагальненого алгоритму роботи робота

Програмні коди реалізації функцій робота

Для реалізації функцій робота, враховуючи значну кількість елементів, обрано для роботи плату Arduino MEGA, програмний код написаний мовою C++. Програма складається з таких функцій:

- Функція покрокового переміщення;
- Функція повороту ліворуч;
- Функція повороту праворуч;
- Функція розміщення передавача в задану точку простору (забирання по закінченню роботи);
- Функція побудови маршруту та переміщення робота до точки розміщення передавача;
- Функція побудови маршруту та переміщення робота в заданій множині точок простору;
- Функція вимірювання відстаней до перешкод;
- Функція оминання перешкод;
- Програма роботи передавача;
- Функція роботи приймача;
- Функція розрахунку та виведення результатів оцінки рівнів Wi-Fi сигналів.

ISSN 2663 - 4023

```
// підключення бібліотек
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Keypad.h>
#include <iarduino_RF433_Receiver.h>
#include <ArduinoSTL.h>
#include <vector>
#include <math.h>

// оголошення LCD-екрану та сервомоторів
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo legA, legB, legC, legD;
Servo kneeA, kneeB, kneeC, kneeD;
Servo hipA, hipB, hipC, hipD;
Servo cell;

// необхідні змінні, а саме: початкові положення моторів, кути нахилу, час,
// виконання кроку, координати точки простору, координати робота, відносна відстань
// між координатами, товщина перешкоди, напрямок робота, буфер очистки та прапорці
int legAStartPosition, legBStartPosition, legCStartPosition, legDStartPosition;
int kneeAStartPosition, kneeBStartPosition, kneeCStartPosition, kneeDStartPosition;
int hipAStartPosition, hipBStartPosition, hipCStartPosition, hipDStartPosition;
int legAngle = 30;
int kneeAngle = 45;
int stepDelay = 1000;
int currentX = 0;
int currentY = 0;
int stepCount = 0;
int deltaX = 0;
int deltaY = 0;
int x = 0;
int y = 0;
int sx = 0;
int sy = 0;
int k = 0;
int newData[];
String input = "";
String obstacle = "false";
String start = "true";
String currentDirection = "forward";
String inputBuffer = "";
iarduino_RF433_Receiver radio(11);
std::vector<int> receivedData;
const int trigPinLeft = 22;  // пін для відправки сигналу лівим датчиком
const int echoPinLeft = 23;  // пін для отримання сигналу лівим датчиком
const int trigPinFront = 24; // пін для відправки сигналу переднім датчиком
const int echoPinFront = 25; // пін для отримання сигналу переднім датчиком
const int trigPinRight = 26; пін для відправки сигналу правим датчиком
const int echoPinRight = 27; пін для отримання сигналу правим датчиком

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {35, 34, 33, 32};
byte colPins[COLS] = {31, 30, 29, 28};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);


void setup() {
         radio.begin();
  radio.setDataRate(i433_1KBPS);
  radio.openReadingPipe(5);
  Serial.begin(9600);
  lcd.begin();
  lcd.backlight();
         lcd.clear();
  lcd.setCursor(0, 0);

  legB.attach(2);
  legD.attach(4);
  legA.attach(6);
  legC.attach(8);
  kneeB.attach(3);
  kneeD.attach(5);
  kneeA.attach(7);
  kneeC.attach(9);
  hipB.attach(36);
  hipD.attach(37);
  hipA.attach(38);
  hipC.attach(39);

  legAStartPosition = legA.read();
  legBStartPosition = legB.read();
  legCStartPosition = legC.read();
  legDStartPosition = legD.read();
  kneeAStartPosition = kneeA.read();
  kneeBStartPosition = kneeB.read();
  kneeCStartPosition = kneeC.read();
  kneeDStartPosition = kneeD.read();
  hipAStartPosition = hipA.read();
  hipBStartPosition = hipB.read();
  hipCStartPosition = hipC.read();
  hipDStartPosition = hipD.read();
// ініціалізація датчиків HC-SR04
  pinMode(trigPinLeft, OUTPUT);
  pinMode(echoPinLeft, INPUT);
  pinMode(trigPinFront, OUTPUT);
  pinMode(echoPinFront, INPUT);
  pinMode(trigPinRight, OUTPUT);
  pinMode(echoPinRight, INPUT);
}
```

Рис. 5. Початкова частина програмного коду

ISSN 2663 - 4023

Функція покрокового переміщення реалізована за допомогою опорно-рухової системи робота, згідно представленої схеми на рис. 6, де A, B, C, D — кінцівки робота, відрізки A-A1, B-B1, C-C1, D-D1 — їх проекції після зробленого кроку, G — центр маси робота, зафарбовані точки — кінцівки на підлозі, незафарбовані — у повітрі. Робот має крокувати таким чином, щоб три його кінцівки (утворюють зафарбований трикутник) стояли на підлозі, а четверта виконувала крок і знаходилась у повітрі. Черговість кроків наступна: B, C, D, A.

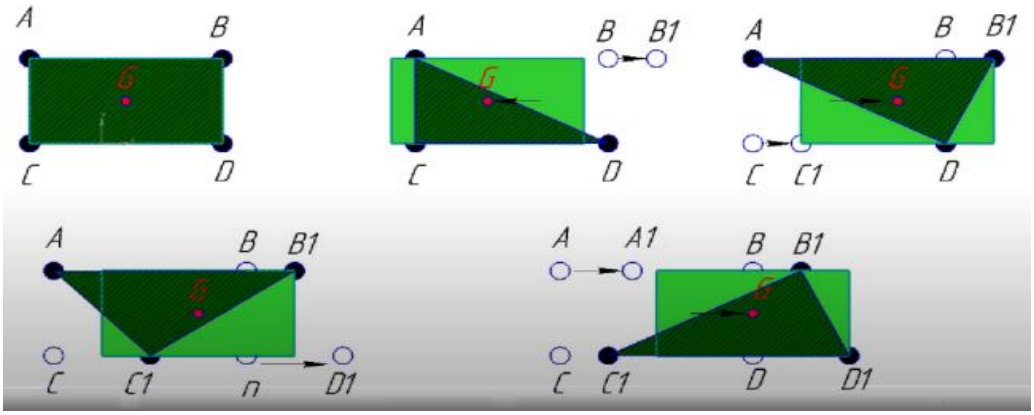

Рис. 6. Схематичне зображення покрокового переміщення робота

Програмний код реалізації даної функції подано на рис. 7.

```
void moveForward() {
  // крок кінцівкою B
  legB.write(legAngle);
  kneeB.write(kneeAngle);
  delay(stepDelay / 4);
  legB.write(-legAngle + 15);
  kneeB.write(-kneeAngle);
  delay(stepDelay / 2);

  // крок кінцівкою C
  legC.write(-legAngle);
  kneeC.write(-kneeAngle);
  delay(stepDelay / 4);
  legC.write(legAngle + 15);
  kneeC.write(kneeAngle);
  delay(stepDelay / 2);
  // крок кінцівкою D

   legD.write(-legAngle);
  kneeD.write(-kneeAngle);
  delay(stepDelay / 4);
  legD.write(legAngle + 15);
  kneeD.write(kneeAngle);
  delay(stepDelay / 2);

  // рух кінцівкою A
  legA.write(legAngle);
  kneeA.write(kneeAngle);
  delay(stepDelay / 4);
  legA.write(-legAngle + 15);
  kneeA.write(-kneeAngle);
  delay(stepDelay / 2);
}
```

Рис. 7. Програмний код реалізації функції покрокового переміщення робота

Функція повороту ліворуч (як приклад, на 45 градусів) реалізована за допомогою відповідних шарнірів опорно-рухової системи робота. Згідно функції робот поверне ліворуч від попереднього напрямку руху на 45 градусів. Програмний код реалізації даної функції подано на рис. 8.

ISSN 2663 - 4023

```
void turnLeft(int rep) {
  for (int i = 0; i < rep; i++) {
//Нахил ліворуч і одночасне підняття кінцівок B і C
    hipA.write(45);
    hipD.write(45);
    legB.write(-legAngle + 15);
    kneeB.write(kneeAngle);
    legC.write(legAngle + 15);
    kneeC.write(-kneeAngle);
//Вирівнювання і одночасне опускання ніг B і C
    hipA.write(-45);
    hipD.write(-45);
    legB.write(legAngle + 15);
    kneeB.write(-kneeAngle);
    legC.write(-legAngle + 15);
    kneeC.write(kneeAngle);
    delay(stepDelay / 2);

//Нахил ліворуч і одночасне підняття ніг A і D
    hipB.write(45);
    hipC.write(45);
    legA.write(-legAngle + 15);
    kneeA.write(kneeAngle);
    legD.write(legAngle + 15);
    kneeD.write(-kneeAngle);
//Вирівнювання і одночасне опускання ніг A і D
    hipB.write(-45);
    hipC.write(-45);
    legA.write(legAngle + 15);
    kneeA.write(-kneeAngle);
    legD.write(-legAngle + 15);
    kneeD.write(kneeAngle);
    delay(stepDelay / 2);
  }
}
```

Рис. 8. Програмний код реалізації функції повороту ліворуч

Функція повороту праворуч (як приклад на 45 градусів) реалізована аналогічно функції повороту ліворуч, але в протилежну сторону відносно напрямку руху. Програмний код реалізації даної функції подано на рис. 9.

```
void turnRight(int rep) {
  for (int i = 0; i < rep; i++) {
//Нахил ліворуч і одночасне підняття ніг A і D
    hipB.write(-45);
    hipC.write(-45);
    legA.write(-legAngle + 15);
    kneeA.write(kneeAngle);
    legD.write(legAngle + 15);
    kneeD.write(-kneeAngle);
//Вирівнювання і одночасне опускання ніг A і D
    hipB.write(45);
    hipC.write(45);
    legA.write(legAngle + 15);
    kneeA.write(-kneeAngle);
    legD.write(-legAngle + 15);
    kneeD.write(kneeAngle);
    delay(stepDelay / 2);

//Нахил ліворуч і одночасне підняття ніг B і C
    hipA.write(-45);
    hipD.write(-45);
    legB.write(-legAngle + 15);
    kneeB.write(kneeAngle);
    legC.write(legAngle + 15);
    kneeC.write(-kneeAngle);
//Вирівнювання і одночасне опускання ніг B і C
    hipA.write(45);
    hipD.write(45);
    legB.write(legAngle + 15);
    kneeB.write(-kneeAngle);
    legC.write(-legAngle + 15);
    kneeC.write(kneeAngle);
    delay(stepDelay / 2);
  }
}
```

Рис. 9. Програмний код реалізації функції повороту праворуч

Функція розміщення передавача (забирання по закінченню роботи) реалізована за допомогою окремого сервомотора для відкриття та закриття транспортного відсіку. Програмний код реалізації даної функції подано на рис. 10.

ISSN 2663 - 4023

```
void Cell() {
          kneeA.write(-kneeAngle);
          kneeC.write(kneeAngle);
          delay(stepDelay / 2);
          kneeB.write(-kneeAngle);
          kneeD.write(kneeAngle);
          delay(stepDelay / 2);
          cell.write(180); // відкриття відсіку
          delay(stepDelay);
          cell.write(-180); // закриття відсіку
          kneeB.write(kneeAngle);
          kneeD.write(-kneeAngle);
          delay(stepDelay / 2);
          kneeA.write(kneeAngle);
          kneeC.write(-kneeAngle);
}
```

Рис. 10. Програмний код реалізації функції розміщення передавача (забирання по закінченню роботи)

Функція зупинки робота реалізована за рахунок запису положення моторів до початку роботи. Програмний код реалізації даної функції подано на рис. 11

```
void stopRobot() {
  legA.write(legAStartPosition);
  legB.write(legBStartPosition);
  legC.write(legCStartPosition);
  legD.write(legDStartPosition);
  kneeA.write(kneeAStartPosition);
  kneeB.write(kneeBStartPosition);
  kneeC.write(kneeCStartPosition);
  kneeD.write(kneeDStartPosition);
  hipA.write(hipAStartPosition);
  hipB.write(hipBStartPosition);
  hipC.write(hipCStartPosition);
  hipD.write(hipDStartPosition);
}
```

Рис. 11. Програмний код реалізації функції зупинки робота

Функція вимірювання відстаней до перешкод реалізується за викликом необхідного датчика, а саме пінів відправки і прийняття сигналу, та подальшого його запуску із отриманням значення часу, за який сигнал подолав відстань до перешкоди. Програмний код реалізації даної функції подано на рис. 12.

```
long measureDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  return pulseIn(echoPin, HIGH);
}
```

Рис. 12. Програмний код реалізації функції вимірювання відстаней до перешкод

Функція оминання перешкод реалізується за викликом функції вимірювання відстаней для подальших розрахунків. Після визначення відстані до перешкоди, залежно від умови, положення робота та відносних координат, приймається певне рішення про оминання перешкоди та корегується маршрут. Програмний код реалізації даної функції подано на рис. 13.

```
void checkObstacles() {
  // змінні для вимірювання відстаней
  long durationLeft, distanceLeft;
  long durationFront, distanceFront;
  long durationRight, distanceRight;

  // розрахунок відстаней за допомогою датчиків
  durationLeft = measureDistance(trigPinLeft, echoPinLeft);
  distanceLeft = durationLeft * 0.034 / 2;
  durationFront = measureDistance(trigPinFront, echoPinFront);
  distanceFront = durationFront * 0.034 / 2;
  durationRight = measureDistance(trigPinRight, echoPinRight);
  distanceRight = durationRight * 0.034 / 2;

  // аналізуємо відстані і приймаємо рішення
  if (distanceFront < 20) {
    stopRobot();
  stepCount = 0;
  obstacle = "true";
    if (distanceLeft < 20) {
      turnRight(2);
      moveForward(1);
      while (distanceLeft < 20) {
        moveForward(1);
              stepCount++;
        durationLeft = measureDistance(trigPinLeft, echoPinLeft);
        distanceLeft = durationLeft * 0.034 / 2;
      }
      turnLeft(2);
          if (currentDirection = "forward") {
                    deltaY--;
                    if (x > currentX) {
                              deltaX = deltaX + stepCount;
                    } else if (x < currentX) {
                              deltaX = deltaX - stepCount;
                    }
          }
          if (currentDirection = "backward") {
                    deltaY--;
                    if (x > currentX) {
                              deltaX = deltaX - stepCount;
                    } else if (x < currentX) {
                              deltaX = deltaX + stepCount;
                    }
          }
          if (currentDirection = "left") {
                    deltaX--;
                    if (y > currentY) {
                              deltaY = deltaY - stepCount;
                    } else if (y < currentY) {
                              deltaY = deltaY + stepCount;
                    }
          }
          if (currentDirection = "right") {
                    deltaX--;
                    if (y > currentY) {
                              deltaY = deltaY + stepCount;
                    } else if (y < currentY) {
                              deltaY = deltaY - stepCount;
                    }
}

} else if (distanceRight < 20) {
      turnLeft(2);
      moveForward(1);
      while (distanceRight < 20) {
        moveForward(1);
              stepCount++;
        durationRight = measureDistance(trigPinRight, echoPinRight);
        distanceRight = durationRight * 0.034 / 2;
      }
      turnRight(2);
          if (currentDirection = "forward") {
                    deltaY--;
                    if (x > currentX) {
                              deltaX = deltaX - stepCount;
                    } else if (x < currentX) {
                              deltaX = deltaX + stepCount;
                    }
}
          if (currentDirection = "backward") {
                    deltaY--;
                    if (x > currentX) {
                              deltaX = deltaX + stepCount;
                    } else if (x < currentX) {
                              deltaX = deltaX - stepCount;
                    }
          }
          if (currentDirection = "left") {
                    deltaX--;
                    if (y > currentY) {
                              deltaY = deltaY + stepCount;
                    } else if (y < currentY) {
                              deltaY = deltaY - stepCount;
                    }
          }
          if (currentDirection = "right") {
                    deltaX--;
                    if (y > currentY) {
                              deltaY = deltaY - stepCount;
                    } else if (y < currentY) {
                              deltaY = deltaY + stepCount;
                    }
          }
}}
```

Рис. 13. Програмний код реалізації функції оминання перешкод

ISSN 2663 - 4023

Функція побудови маршруту та переміщення до точки розміщення передавача реалізується після введення користувачем пари координат точки простору, яка оброблюється та прокладається маршрут до точки розміщення передавача. Паралельно функціонує програма виявлення перешкод для вчасної зупинки та їх оминання. Програмний код реалізації даної функції подано на рис. 14.

```
void Start(String startinput) {
          lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Move to start");
  for (int i = 0; i < input.length(); i += 4) {
    x = input.substring(i, i + 2).toInt();
    y = input.substring(i + 2, i + 4).toInt();

                  sx = x; // запис координат знаходження передавача
                  sy = y;
    deltaX = abs(x - currentX);
    deltaY = abs(y - currentY);

         while (!(currentX == x && currentY == y)) {
         checkObstacles();
         }

    if (x > currentX && y>currentY) {
      if (currentDirection == "forward") {
                              turnRight(2);
                              currentDirection = "right";
                      }
      if (currentDirection == "backward") {
                              turnLeft(2);
                              currentDirection = "right";
                      }
      if (currentDirection == "right") {
                              currentDirection = "right";
                      }
      if (currentDirection == "left") {
                              turnRight(4);
                              currentDirection = "right";
                      }
      moveForward(deltaX);
      turnLeft(2);
                              currentDirection = "forward";
      moveForward(deltaY);
    }
    if (x > currentX && y<currentY) {
      if (currentDirection == "forward") {
                              turnRight(2);
                              currentDirection = "right";
                      }
      if (currentDirection == "backward") {
                              turnLeft(2);
                              currentDirection = "right";
                      }
      if (currentDirection == "right") {
                              currentDirection = "right";
                      }
      if (currentDirection == "left") {
                              turnRight(4);
                              currentDirection = "right";
                      }
      moveForward(deltaX);
      turnRight(2);
                              currentDirection = "backward";
      moveForward(deltaY;
    }
    if (x < currentX && y>currentY) {
      if (currentDirection == "forward") {
                              turnLeft(2);
                              currentDirection = "left";
                      }
      if (currentDirection == "backward") {
                              turnRight(2);
                              currentDirection = "left";
                      }
      if (currentDirection == "right") {
                              turnRight(4);
                              currentDirection = "left";
                      }
      if (currentDirection == "left") {
                              currentDirection = "left";
                      }
      moveForward(deltaX);
      turnRight(2);
                              currentDirection = "forward";
      moveForward(deltaY);
    }
    if (x < currentX && y<currentY) {
      if (currentDirection == "forward") {
                              turnLeft(2);
                              currentDirection = "left";
                      }
      if (currentDirection == "backward") {
                              turnRight(2);
                              currentDirection = "left";
                      }
      if (currentDirection == "right") {
                              turnRight(4);
                              currentDirection = "left";
                      }
      if (currentDirection == "left") {
                              currentDirection = "left";
                      }
      moveForward(deltaX);
      turnLeft(2);
                              currentDirection = "backward";
      moveForward(deltaY);
    }

    currentX = x;
    currentY = y;
  }
}
```

Рис. 14. Програмний код реалізації функції побудови маршруту та переміщення до точки розміщення передавача

ISSN 2663 - 4023

Функція побудови маршруту та переміщення в заданій множині точок простору реалізується після введення користувачем координат точок простору, які оброблюються та прокладається маршрут. Паралельно програма очікує зустріч із перешкодою для вчасної зупинки та її обходу. Програмний код реалізації даної функції подано на рис. 15.

```
void processInput(String input) {
            lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Work in progress");
  for (int i = 0; i < input.length(); i += 4) {
    x = input.substring(i, i + 2).toInt();
    y = input.substring(i + 2, i + 4).toInt();

    deltaX = abs(x - currentX);
    deltaY = abs(y - currentY);

           while (!(currentX == x && currentY == y)) {
           checkObstacles();
           }
    if (x > currentX && y>currentY) {
      if (currentDirection == "forward") {
                                turnRight(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "backward") {
                                turnLeft(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "right") {
                                currentDirection = "right";
                        }
      if (currentDirection == "left") {
                                turnRight(4);
                                currentDirection = "right";
                        }
      moveForward(deltaX);
      turnLeft(2);
                                currentDirection = "forward";
      moveForward(deltaY);
    }
    if (x > currentX && y<currentY) {
      if (currentDirection == "forward") {
                                turnRight(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "backward") {
                                turnLeft(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "right") {
                                currentDirection = "right";
                        }
      if (currentDirection == "left") {
                                turnRight(4);
                                currentDirection = "right";
                        }
      moveForward(deltaX);
      turnRight(2);
                                currentDirection = "backward";
      moveForward(deltaY;
    }


void processInput(String input) {
            lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Work in progress");
  for (int i = 0; i < input.length(); i += 4) {
    x = input.substring(i, i + 2).toInt();
    y = input.substring(i + 2, i + 4).toInt();

    deltaX = abs(x - currentX);
    deltaY = abs(y - currentY);

           while (!(currentX == x && currentY == y)) {
           checkObstacles();
           }
    if (x > currentX && y>currentY) {
      if (currentDirection == "forward") {
                                turnRight(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "backward") {
                                turnLeft(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "right") {
                                currentDirection = "right";
                        }
      if (currentDirection == "left") {
                                turnRight(4);
                                currentDirection = "right";
                        }
      moveForward(deltaX);
      turnLeft(2);
                                currentDirection = "forward";
      moveForward(deltaY);
    }
    if (x > currentX && y<currentY) {
      if (currentDirection == "forward") {
                                turnRight(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "backward") {
                                turnLeft(2);
                                currentDirection = "right";
                        }
      if (currentDirection == "right") {
                                currentDirection = "right";
                        }
      if (currentDirection == "left") {
                                turnRight(4);
                                currentDirection = "right";
                        }
      moveForward(deltaX);
      turnRight(2);
                                currentDirection = "backward";
      moveForward(deltaY;
    }
```

Рис. 15. Програмний код реалізації функції побудови маршруту та переміщення в заданій множині точок простору

Програмний код передавача розміщується в самому передавачі (Arduino UNO), оскільки він розташовується окремо від роботизованої платформи. Програмний код реалізації даної функції подано на рис. 16.

```
#include <iarduino_RF433_Transmitter.h>
iarduino_RF433_Transmitter radio(11); // під'єднуємо передавач до піна 11
int data[1];  // створюємо масив передачі даних

void setup() {
  radio.begin();  // ініціалізуємо роботу передавача
  radio.setDataRate(i433_1KBPS); // встановлюємо швидкість передачі даних
  radio.openWritingPipe(5);  // відкриваємо канал для передачі даних
}

void loop() {
  data[0] = random(0, 16);
  radio.write(&data, sizeof(data)); // відправляємо дані з масива
  delay(2000);  // пауза перед наступною відправкою
}
```

Рис. 16. Програмний код роботи передавача

Функція роботи приймача. Сигнал приймача фіксується в момент досягнення роботом поточної точки простору. Програмний код реалізації даної функції подано на рис. 17.

```
void receiveDataOnce() {
      radio.startListening();          // починаємо моніторинг каналу

    // очікуємо наявність даних на протязі секунди
    unsigned long startTime = millis(); // Запам'ятовуємо час очікування
    while (!radio.available() && (millis() - startTime) < 1000) {
        delay(10);             // пауза між перевірками доступності даних
    }

    if (radio.available()) {          // в разі наявності доступних даних
        newData[k] = 1;          // в разі наявності доступних даних
    } else { newData[k] = 0;
    }

    radio.stopListening();            // завершуємо моніторинг каналу
}
```

Рис. 17. Програмний код реалізації функції роботи приймача

Функція розрахунку та виведення результатів оцінки рівнів Wi-Fi сигналів. На екран виводиться множина точок простору із недостатнім рівнем Wi-Fi сигналу. Програмний код реалізації даної функції подано на рис. 18.

```
Calculating(){
bool foundZero = false; // прапор, що вказує на наявність точки недостатньої потужності в масиві

  // Перевірка кожного елемента масива
  for (int i = 0; i < k; i++) {
    // Якщо елемент масива дорівнює 0, виводимо його номер на екран
    if (newData[i] == 0) {
      foundZero = true; // встановлюємо пропор наявності 0 в масиві
      lcd.print("Points with zero signals: "); // виводимо текст
      lcd.print(i + 1); // виводимо номер точки простору
      if (i < arraySize - 1 && myArray[i + 1] == 0) {
        lcd.print(", "); // виведення коми
      }
    }
  }
  // якщо точка недостатньої потужності відсутня, то виводимо повідомлення
  if (!foundZero) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Signal is good");
  }}
```

Рис. 18. Програмний код реалізації функції розрахунку та виведення результатів оцінки рівнів Wi-Fi сигналів

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Запропонована методика швидкої оцінки потужності Wi-Fi-сигналу в точках простору, відокремлених перешкодами з різними коефіцієнтами згасання, характеризується низькими обчислювальними витратами, що з успіхом може бути використано для оптимізації розташування точок доступу в межах приміщення із заданою геометрією і забезпечення стійкого покриття простору приміщення Wi-Fi-сигналом.

Подальший технічний прогрес у створенні більш досконалої електроніки дозволить здійснити модифікацію робота для підвищення точності орієнтування у просторі і точності оцінки потужності Wi-Fi-сигналу, що підвищить ефективність його застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шокало, В. М., Усін, В. А., Грецьких, Д. В., Хорошко, В. О., & Крючкова, Л. П. (2012). *Поля і хвилі в системах технічного захисту інформації: підручник для студентів вищих навчальних закладів*. *Ч.1*. Колегіум.
2. Jivthesh, M. R, Gaushik, M. R, Heshan Niranga, G. D., & Punathil, A. (2022). A Comprehensive survey of WiFi Analyzer Tools. *IEEE 3rd Global Conference for Advancement in Technology (GCAT).* https://doi.org/10.1109/GCAT55367.2022.9972040
3. Fredriksson, S. (2021). *Design, Development and Control of a Quadruped Robot*.
4. Spot | Boston Dynamics. (б.д.). *Boston Dynamics*. https://bostondynamics.com/products/spot/
5. MaxBotix. (б.д.). *How Ultrasonic Sensors Work.* https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work#:~:text=What%20is%20an%20Ultrasonic%20Sensor,information%20about%20an%20object's%20proximity
6. Arduino - Home. (б.д.). *Arduino Mega ADK Rev3 - Arduino Reference.* https://docs.arduino.cc/retired/boards/arduino-mega-adk-rev3/?queryID=undefined
7. Arduino - Home. (б.д.). *HC_SR04 - Arduino Reference.* https://www.arduino.cc/reference/en/libraries/hc_sr04/
8. Arduino - Home. (б.д.). *LCD_I2C - Arduino Reference.* https://www.arduino.cc/reference/en/libraries/lcd_i2c/
9. Arduino - Home. (б.д.). *MultitapKeypad - Arduino Reference.* https://www.arduino.cc/reference/en/libraries/multitapkeypad/
10. Безпроводовий зв'язок для Arduino на 433Мгц (FS1000A и MX-RM-5V). (2018). *РадіоЛис - Інтернет-журнал.* https://radiolis.pp.ua/arduino/49-besprovodnaya-svyaz-arduino-fs1000a-mx-rm-5v

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (24), 2024
ISSN 2663 - 4023

Larysa Kriuchkova
Doctor of sciences, professor, professor of Volodymyr Buriachok
Department of Information and Cybersecurity
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID: 0000-0002-8509-6659
l.kriuchkova@kubg.edu.ua

Nikita Leontiuk
student
Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
nsleontiuk.fitu20@kubg.edu.ua

SOFTWARE AND HARDWARE IMPLEMENTATION OF THE ALGORITHM FOR QUICK ASSESSMENT OF WI-FI SIGNAL POWER AT POINTS OF THE URBANIZED SPACE

Abstract. In connection with the widespread use of Wireless Fidelity wireless technologies; there is an urgent problem of ensuring the proper level of Wi-Fi signal in the space of urbanized premises. The presence of walls, partitions, furniture, radio-electronic equipment and other objects inside the building complicates the conditions for the propagation of radio waves. The main effects observed during the propagation of radio waves indoors are multipath due to multiple reflections of radio waves from walls and other objects, diffraction at numerous sharp edges of objects, and attenuation of radio waves during propagation and when passing through obstacles. For the effective use of wireless networks in the specified conditions, it is necessary to be able to quickly assess the Wi-Fi signal level in the space of the room. The purpose of the publication is the software-hardware implementation of the algorithm for quick assessment of the Wi-Fi signal strength in multiple points of the space of an urbanized space. A version of a robotic platform with the necessary electronics and software is presented, capable of automatically performing a quick assessment of the Wi-Fi signal strength at a given set of points in an urbanized space; block diagram of the generalized algorithm of the robot; software codes for implementations of robot functions and the Wi-Fi signal strength estimation algorithm. The proposed technique for quick assessment of the Wi-Fi signal strength at points in space separated by obstacles with different attenuation coefficients is characterized by low computational costs, which can be successfully used to optimize the location of access points within a room with a given geometry and ensure stable coverage of the space of the room Wi- Fi signal. Further technical progress in the creation of more advanced electronics will make it possible to modify the robot to increase the accuracy of orientation in space and the accuracy of estimating the strength of the Wi-Fi signal, which will increase the efficiency of its use.

Keywords: wireless technologies; assessment of Wi-Fi signal strength; robotic platform; urbanized premises; set of points in space; algorithm; program code.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Shokalo, V. M., Usin, V. A., Gretskikh, D. V., Khoroshko, V. O., & Kryuchkova, L. P. (2012*). Fields and waves in systems of technical protection of information: a textbook for students of higher educational institutions, Ch. 1.* Collegium.
2. Jivthesh, M. R, Gaushik, M. R, Heshan Niranga, G. D., & Punathil, A. (2022). A Comprehensive survey of WiFi Analyzer Tools. *IEEE 3rd Global Conference for Advancement in Technology (GCAT).* https://doi.org/10.1109/GCAT55367.2022.9972040
3. Fredriksson, S. (2021). *Design, Development and Control of a Quadruped Robot*.
4. Spot | Boston Dynamics. (n.d.). *Boston Dynamics*. https://bostondynamics.com/products/spot/
5. MaxBotix. (n.d.). *How Ultrasonic Sensors Work.* https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work#:~:text=What%20is%20an%20Ultrasonic%20Sensor,information%20about%20an%20object's%20proximity

6. Arduino - Home. (n.d.). *Arduino Mega ADK Rev3 - Arduino Reference*. https://docs.arduino.cc/retired/boards/arduino-mega-adk-rev3/?queryID=undefined
7. Arduino - Home. (n.d.). *HC_SR04 - Arduino Reference*. https://www.arduino.cc/reference/en/libraries/hc_sr04/
8. Arduino - Home. (n.d.). *LCD_I2C - Arduino Reference*. https://www.arduino.cc/reference/en/libraries/lcd_i2c/
9. Arduino - Home. (n.d.). *MultitapKeypad - Arduino Reference*. https://www.arduino.cc/reference/en/libraries/multitapkeypad/
10. Wireless communication for Arduino at 433MHz (FS1000A and MX-RM-5V). (2018). *RadioLis - an online magazine*. https://radiolis.pp.ua/arduino/49-besprovodnaya-svyaz-arduino-fs1000a-mx-rm-5v