**Дніпровський ліцей інформаційних технологій**

**при Дніпровському національному університеті**

**імені Олеся Гончара**

**Випускна робота**

**на тему:**

**«Система трекінгу переміщення визначеного об’єкта»**

**Виконавець:**

**ліцеїст 11-В-2 класу**

**Федоренко Дмитро**

**Науковий керівник:**

**Чашка Ю.М. \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дніпро**

**2018**

ЗМІСТ

[ВСТУП 3](#_Toc530805354)

[ОСНОВНА ЧАСТИНА 4](#_Toc530805355)

[ТЕОРЕТИЧНА ЧАСТИНА 4](#_Toc530805356)

[МЕТОД ЗАМІРІВ 4](#_Toc530805357)

[ЕЛЕМЕНТИ КОНСТРУКЦІЇ 6](#_Toc530805358)

[СХЕМА МОДЕЛІ 7](#_Toc530805359)

[ЗБІР ДАНИХ 8](#_Toc530805360)

[ПРОГРАМНА ЧАСТИНА 9](#_Toc530805361)

[ВЗАЄМОДІЯ КОМП’ЮТЕРНОГО ПРОЕКТУ ТА ARDUINO 9](#_Toc530805362)

[ЕЛЕМЕНТ SERIAL PORT 9](#_Toc530805363)

[РОЗПОДІЛЕННЯ ПРОЦЕСІВ МІЖ ПРОЕКТАМИ 11](#_Toc530805364)

[СИНХРОНІЗАЦІЯ ДАНИХ 12](#_Toc530805365)

[РОЗРАХУНОК ПОТРІБНИХ ПАРАМЕТРІВ 16](#_Toc530805366)

[ВИВІД РЕЗУЛЬТАТІВ КОРИСТУВАЧУ 17](#_Toc530805367)

[ПОСІБНИК КОРИСТУВАЧА 18](#_Toc530805368)

[СТРУКТУРНА СХЕМА 19](#_Toc530805369)

[ВИСНОВКИ 20](#_Toc530805370)

# ВСТУП

**Актуальність** дослідження полягає у створенні засобу та технології стеження за об’єктом, які, на відміну від сучасних реалізацій будуть придатні до роботи з меншими затратами. Адже, на даний момент існуючі реалізації передбачають програмний аналіз фотознімків, виконаних за допомогою відеокамери. Тобто цей механізм потребує значних ресурсних та матеріальних вкладень у дослідження.

**Мета роботи** – створити прилад на основі Arduino та комп’ютерні програмні засоби, одним з яких є комп’ютерний проект для демонстрації роботи системи трекінгу, та за допомогою якого можна визначати параметри розташування об’єкту або відомості про теоретичні методи трекінгу.

Виходячи з мети, було поставлено наступні задачі:

1. Прилад
   1. Доступний
   2. Технологічно простий
   3. Зручний для користування
   4. Придатний до подальшого розвитку
2. Комп’ютерний проект
   1. Синхронізація режимів роботи комп’ютерного проекту та мікроконтролера
   2. Програмний елемент-демонстрація трекінгу

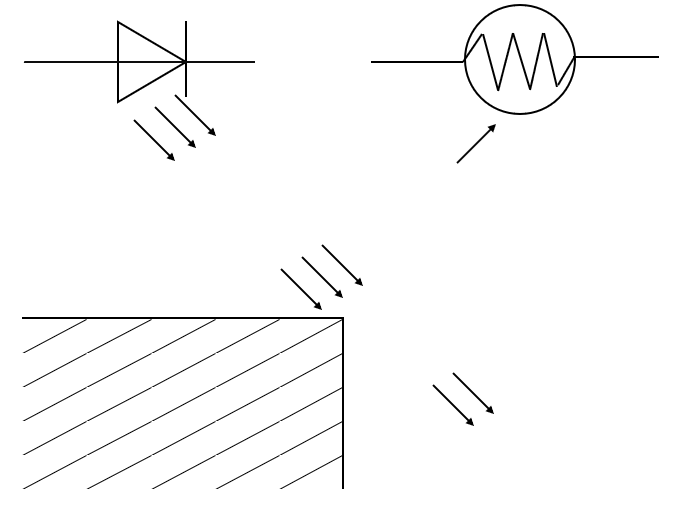
Теоретичне обґрунтування використаних властивостей та реалізованих функцій

# ОСНОВНА ЧАСТИНА

## ТЕОРЕТИЧНА ЧАСТИНА

### МЕТОД ЗАМІРІВ

В основу дослідження покладено визначення переміщення об’єкта за допомогою знаходження різниці рівнів заміряного світла, відбитого від ділянок його поверхні.

Розглянемо цей метод більш детально:

*мал. 1 «Відбивання світла від поверхні»*

Тобто, при засвітці грані об’єкта частина світла, що потрапило на його поверхню, відбивається, а решта пролітає далі без відбиття, а далі може чи відбиватися від фонових об’єктів з сильним розсіюванням та незначною мірою впливати на замірюємі показання. Таким чином: чим більша частина поверхні об’єкту попадає під поток світла від світлодіода – тим більша його частина відбивається від неї, і, як наслідок, більше світла фіксується фоторезистором. На основі заміряних рівнів від кожної з граней об’єкта можна вирахувати зміну положення об’єкта по кожній з трьох координатних осей.

При розгляданні можливих до застосування методів був інший можливий варіант реалізації дослідження – дослідження рівня відбитого від граней світла з розбивкою на спектри. Суть методу полягала в наступному: Як діючі елементи застосовуються чотири світлодіоди та чотири фоторезистори, кожен з яких має світловий фільтр. Фіксуватися рівень відбитого світла фоторезисторами буде в конкретному кольоровому спектрі. Перевага методу – вища швидкодія, за рахунок того, що світлодіоди працюють одночасно та заміри, що проводяться за менший період часу, ніж той, що витрачається на процес розгоряння світлодіода. Недоліками методу є: низька точність, через те, що при розбивці світла на спектри неможливо відфільтрувати “паразитне” засвічення ділянок в близьких спектрах, таких як, наприклад – при жовтому світловому фільтрі засвіченими опиняться фоторезистори, що відповідальні за червоний і зелений спектри, а червоним та зеленим світлодіодами – фоторезистор жовтого спектру. До того ж, цей метод потребує більшої елементної бази, адже додаткові фоторезистори потрібно задіяти, а це викликає підвищення вартості та ускладнення практичної реалізації. І, як додатковий мінус, будь який колір можна утворити із застосуванням лише трьох кольорів – червоний, синій, зелений, а отже, навіть при ідеально спектрах, неможливо отримати більше трьох досліджуваних областей, тобто це викликає за собою послідовний режим роботи світлодіодів, що в свою чергу нівелює перевагу даного методу дослідження. Тож, цей варіант був відкинутий на користь вище зазначеної альтернативи.

Таким чином, отримано доступний метод вимірювання (діючі елементи – світлодіоди та фоторезистор), проте більш точний, ніж альтернативний йому варіант дослідження відбитого світла з розбивкою на спектри. До того ж, що важливо, метод є дистанційним, тобто не потребує тактильної взаємодії з предметом стеження, ця особливість значно розширює сферу можливих застосувань приладу.

Отже, обраний метод дозволяє реалізувати стеження за переміщенням об’єкту в 3 координатах: X (вліво, вправо), Y (вгору, вниз), Z (вперед, назад).

### ЕЛЕМЕНТИ КОНСТРУКЦІЇ

При розробці моделі конструкції обрані були наступні компоненти:

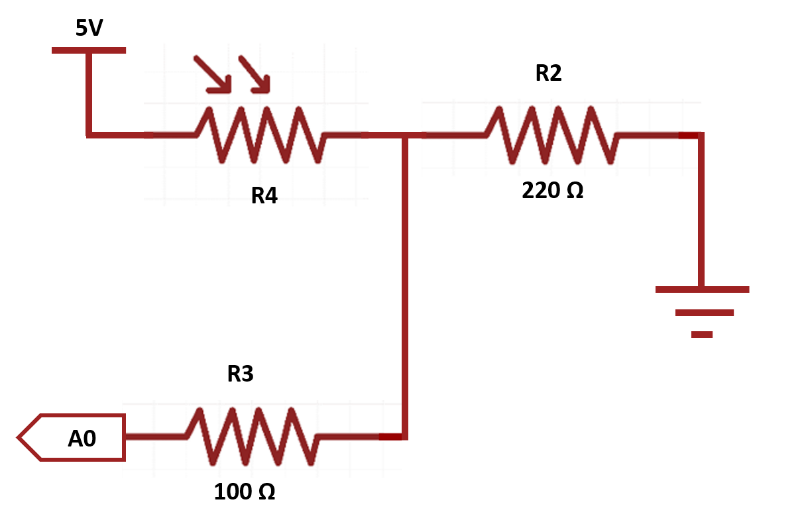
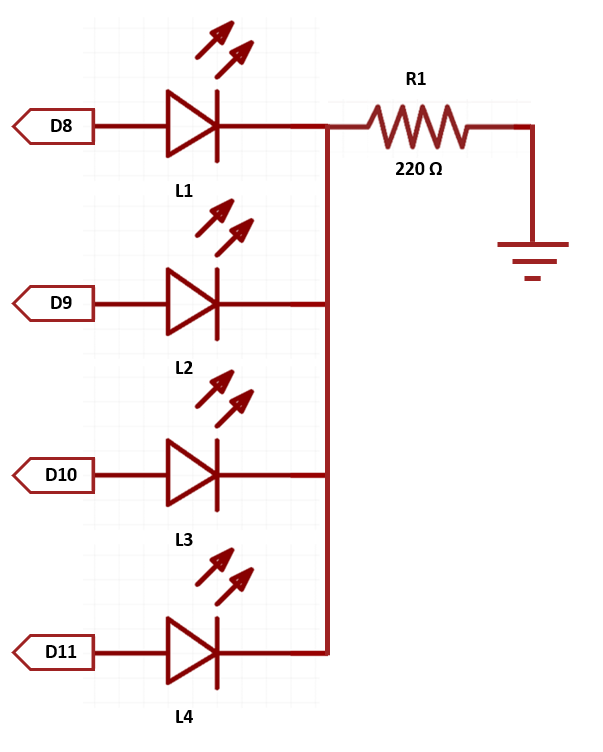
1. Світлодіод – напівпровідниковий пристрій, що, при пропусканні через нього електричного струму випромінює світло. На відміну від ламп розжарювання, світлодіоди випромінюють світло певної довжини хвилі і в певному напрямі. Остання особливість є доволі важливою, адже для ефективного дослідження ми маємо можливість отримати чітку зону освітлення, що дозволяє підвищити точність вимірювання.
2. Фоторезистор – фотоелектричний напівпровідниковий приймач випромінювання, дія якого ґрунтується на ефекті фотопровідності – явищі зменшення опору напівпровідника у разу збудження носіїв світлом.

*мал. 2 «Світлодіод та фоторезистор»*

Також, додатково використовуються елементи керування – кнопки, за допомогою яких реалізовані функції паузи та зміни центральної точки при роботі програми.

Зазначені вище компоненти мають вагомі переваги, основними з яких є доступність (порівняно низька ціна), достатня точність (можливість виділити конкретизований діапазон дослідження) та відносна простота застосування (легко встановити або замінити).

### СХЕМА МОДЕЛІ

 Структурно модель вимірювального апарату пристрою складають чотири світлодіоди та один фоторезистор, які, в свою чергу, підключені до пінів мікроконтролера, що задає керуючі сигнали та проводить збір даних, їх первинну обробку та передачу у комп’ютер.

*Схема 1 «Електронні схеми приладу»*

При розробці схеми до неї були висунуті такі вимоги: вона має бути компактною, надійною та ефективною, але задля цього не можна жертвувати функціоналом.

Тож у фінальній реалізації схему складають два електричних кола: окремо лінія живлення та керування світлодіодами та окремо лінія роботи з фоторезистором. Це зроблено з метою підвищення точності замірів, адже струм, що протікає через світлодіоди не торкає фоторезистор, а струм з фоторезистора не живить світлодіоди, отже ми цією дією отримуємо рівномірну підсвітку світлодіодів та стабільний струм у фоторезисторі, що мінімізує можливу похибку у вимірювннях.

### ЗБІР ДАНИХ

Збір даних з однієї ділянки виконується за наступним алгоритмом:

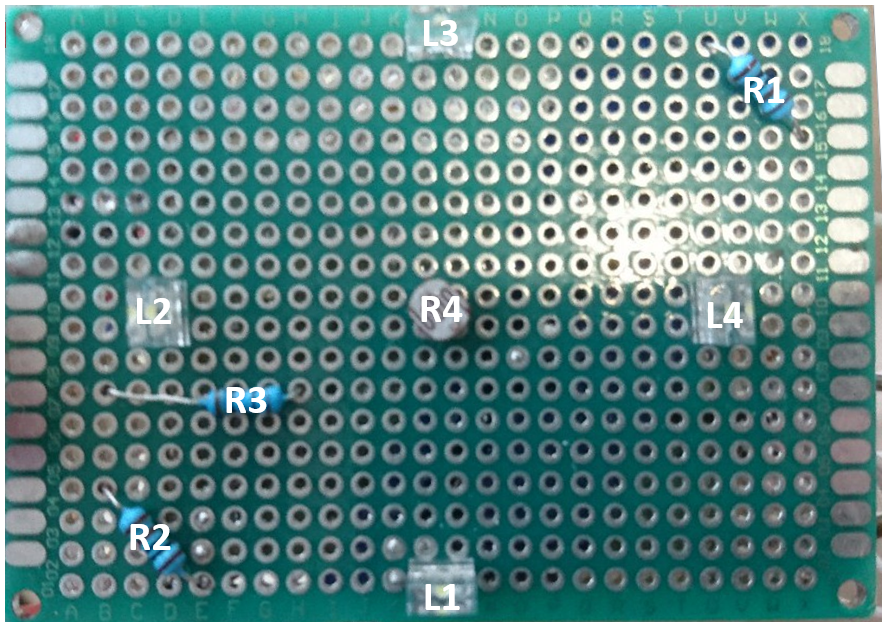
back = analogRead(A0); //збір показників фонового освітлення

digitalWrite(8, 1); //подача струму на світлодіод

    delay(50); //затримка для розгоряння світлодіода

   b = analogRead(A0); //збір показників при ввімкненому світлодіоді

    digitalWrite(8, 0); //вимкнення світлодіоду

Збір даних відбивання світла від граней об’єкта відбувається послідовно: спочатку світлодіод L1, що відповідає нижній грані об’єкта, потім – L2 (лівий), L3 (верхній) та L4 (правий) відповідно. Після збору даних відбувається їх первинна обробка: вираховування різниці отриманих даних та фонового рівня освітлення, а потім – передача отриманих результатів у комп’ютер.

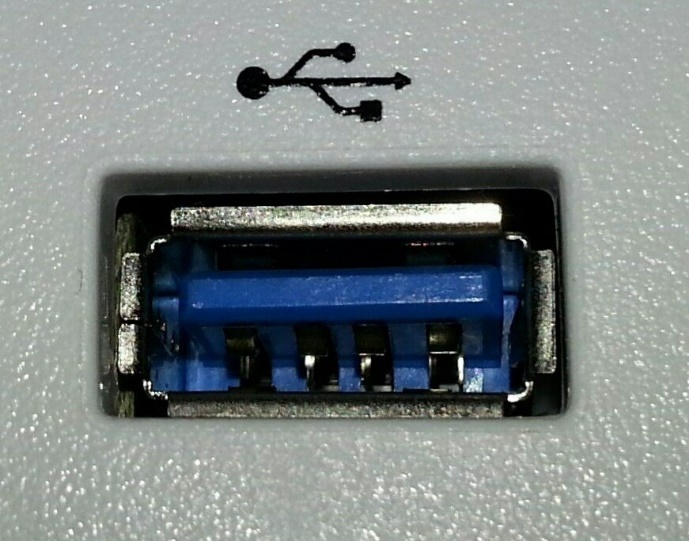
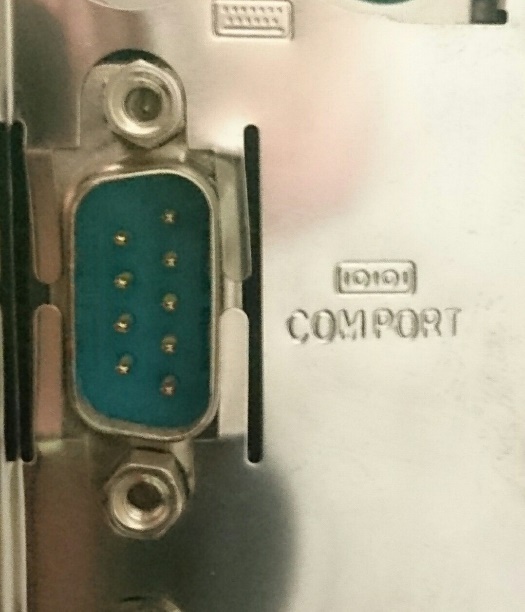
*мал. 3 Макетна схема приладу*

# ПРОГРАМНА ЧАСТИНА

## ВЗАЄМОДІЯ КОМП’ЮТЕРНОГО ПРОЕКТУ ТА ARDUINO

### ЕЛЕМЕНТ SERIAL PORT

Послідовний порт - двонаправлений послідовний аналоговий інтерфейс, призначений для обміну байтовою інформацією. Послідовний тому, що інформація через нього передається по одному біту, біт за бітом.

Наразі послідовні порти фактично вийшли з користування на користь універсального USB, у якому реалізована програмна емуляція COM порта та дані, на відміну від оригіналу, передаються в цифровому вигляді.

*мал. 3 «COM та USB порти»*

Тож, SerialPort – елемент, що дозволяє працювати з USB-контролером з послідовною передачею даних по COM-інтерфейсу.

Використання SerialPort в програмі для мікроконтролера:

Serial.begin(9600); //оголошення SerialPort при значенні BaudRate = 9600

Serial.println(""); //вивід строкового значення до SerialPort

Використання SerialPort в програмі C#:

serialPort1.PortName = "COM?"; //задавання послідовного порту для

//елементу serialPort1

serialPort1.BaudRate = 9600; //задавання параметра BaudRate = 9600

serialPort1.Open(); //відкривання порту

string s = serialPort1.ReadLine(); //задавання строці s значення,

//зчитаного з SerialPort

### РОЗПОДІЛЕННЯ ПРОЦЕСІВ МІЖ ПРОЕКТАМИ

При розгляді питання про синхронізацію проектів виникла додаткова потреба – розподіл виконуваних задач між проектами Arduino та C#, а саме – виконання обчислень. Проводити їх на мікроконтролері чи в комп’ютерному проекті. Переважна більшість виконуваних обчислень була віддана комп’ютерному проекту, через зручність програмування та доступу до кінцевих та проміжних результатів.

В обраній реалізації мікроконтролер виступає в більшій мірі засобом введення, тобто його задача – зібрати дані та передати їх до комп’ютера. Проте, з метою зменшення об’єму даних, що посилаються через послідовний порт на мікроконтролері вираховується згадана вище різниця рівнів фонового освітлення та підсвіченої ділянки. Також, для запобігання конфліктів даних при передачі їх одразу в дві сторони, мікроконтролер відповідальний за функцію паузи спостереження.

    if(digitalRead(3) == HIGH){ //перевірка натиснення кнопки

        pause = !pause; //встановлення змінної у положення false

        while(digitalRead(3) == HIGH); //очікування відпускання кнопки

    }

Для підвищення комфорту користувача, кнопка для перезапису опорної точки об’єкта також була поміщена на блок з мікроконтролером та передана під його контроль.

if(digitalRead(6) == HIGH){ //перевірка натиснення кнопки

        Serial.println("d"); //відсилання сигнального символу

//до послідовного порту

        while(digitalRead(6) == HIGH); //очікування відпускання кнопки

    }

Прийом цього символу в комп’ютерному проекті:

if(s == "d\r") //перевірка наявності цього символу

{

dot = true; //встановлення ключ-змінної до дійсного значення

}

int n = int.Parse(s);

/\*....................................................................\*/

if (k == 0) //перевірка порядкового номеру прийнятого символа

{

db = n; //встановлення поточного значення різниці фіксованого світла

if (b0 == (-1000000) || dot) /\*перевірка на перший прийом даних або

прийом команди перезапису опорної точки\*/

b0 = n; //встановлення поточного значення в якості опорного

}

else

{

if (k == 1)

{

dl = n;

if (l0 == (-1000000) || dot)

l0 = n;

}

else

{

if (k == 2)

{

dt = n;

if (t0 == (-1000000) || dot)

t0 = n;

}

else

{

if (k == 3)

{

dr = n;

if (r0 == (-1000000) || dot)

{

r0 = n;

dot = false; //після закінчення проходу встановлення

} //ключ-змінної до недійсного значення

}

}

}

}

k++;

## СИНХРОНІЗАЦІЯ ДАНИХ

При передачі даних через послідовний порт використовується метод їх синхронізації шляхом передачі двох додаткових символів, а саме: “o” та “c” – вони позначають початок та кінець блоку даних відповідно. Допоки не буде зчитаний символ “o”, дані не почнуть перезаписуватися. Також допоки не буде отримано символ “c” дані не почнуть оброблюватись та виводитись користувачу. Що є добрим – це те, що дані від мікроконтролера передаються одночасно усі після замірів, це мінімізує можливість зупинки програми шляхом несподіваного розриву потоку даних. До того ж це дає достатньо простору комп’ютерному проекту для обробки даних та їх виводу користувачу.

    Serial.println("o");

    Serial.println(db);

    Serial.println(dl);

    Serial.println(dt);

    Serial.println(dr);

    Serial.println("c");

//передача даних суцільним “блоком” одномоментно

Для того, щоб отримувати дані, що надходять через послідовний порт, у реальному часі в комп’ютерному проекті був реалізований програмний фоновий потік, задача якого зчитувати дані, враховуючи поставлені умови синхронізації, оновляти за цими даними програмні змінні та регулювати роботу таймера по розрахункам та графічному відображенню результатів, шляхом указування на те, чи дані оновлені та придатні до обробки, чи ні.

void thr()

{

//На випадок декількох пристроїв, що підключені через послідовний порт та //призодять до помилки в програмі

//serialPort1.PortName = "COM?";

//serialPort1.BaudRate = 9600;

//serialPort1.Open();

string[] port = SerialPort.GetPortNames();

foreach (string p in port) //автопошук потрібного COM-потру

serialPort1.PortName = p;

serialPort1.BaudRate = 9600;

try

{

serialPort1.Open();

}

catch (Exception) { }

bool gate = false; //змінна, що сигналізує чи приходять дані

bool dot = false; //змінна, що сигналізує про зміну опорної точки

int k = 0;

string s = ""; //змінна, в яку будуть записуватись дані із SerialPort

while (true)

{

if (!playing) //умова, що перевіряє чи не показується записаний матеріал

{

if (serialPort1.IsOpen)

{

try

{

s = serialPort1.ReadLine();

}

catch (Exception) { }

if (s != "")

{

if(s == "d\r") //перевірка умови зміни опорної точки

{

dot = true;

}

if (s == "o\r") //початок блоку даних

{

gate = true;

}

if (s == "c\r") //кінець блоку наних

{

gate = false;

updated = true;

k = 0;

}

if (gate)

{

updated = false;

try

{

int n = int.Parse(s);

if (k == 0)

{

db = n;

if (b0 == (-1000000) || dot) /\*перевірка на перший прийом даних або прийом команди перезапису опорної точки\*/

b0 = n; /\*встановлення поточного значення в

якості опорного\*/

}

else

{

if (k == 1)

{

dl = n;

if (l0 == (-1000000) || dot)

l0 = n;

}

else

{

if (k == 2)

{

dt = n;

if (t0 == (-1000000) || dot)

t0 = n;

}

else

{

if (k == 3)

{

dr = n;

if (r0 == (-1000000) || dot)

{

r0 = n;

dot = false;

}

}

}

}

}

k++;

}

catch (Exception)

{

}

}

else

{

}

}

}

}

}

}

### РОЗРАХУНОК ПОТРІБНИХ ПАРАМЕТРІВ

Розрахунок керуючих параметрів відбувається в таймері, та триває в два етапи для кожного з режимів роботи:

1. Розрахунок зміни даних для кожної з чотирьох зон заміру відносно початкового положення

int deltab = frmTitle.db - frmTitle.b0;

int deltal = frmTitle.dl - frmTitle.l0;

int deltat = frmTitle.dt - frmTitle.t0;

int deltar = frmTitle.dr - frmTitle.r0;

// розрахунок для режиму XY

1. Розрахунок загального зміщення об’єкту по осям

int delah = deltab - deltat;

int delaw = deltar - deltal;

// розрахунок для режиму XY

Крім цього існує механізм пост обробки завчасно отриманих та збережених даних, що реалізує моделювання одразу в трьох осях (мається на увазі можливість продивитися переміщення в площині XY та по осі Z за один період часу окремо), що не реалізоване в режимі роботи в реальному часі.

Для цього режиму використовується механізм запису даних, на основі яких будуються графіки до текстового файлу у вихідному форматі. Ці дані, за потреби користувача можуть бути зчитані та відіграні у тому ж порядку, що й поступають до комп’ютерного проекту.

private void timerPW\_Tick(object sender, EventArgs e)

{

if (updated)

{

if (record) //перевірка режиму запису

{

if (k == 0) //перевірка чи перший раз дані поступають

{

File.Delete(Application.StartupPath + "/Database.txt");

//видалення минулої версії файлу

data[0] = "s"; //введення символу, що позначає опорний блок даних

data[1] = b0.ToString();//зповнення масиву опорними даними

data[2] = l0.ToString();//зповнення масиву опорними даними

data[3] = t0.ToString();//зповнення масиву опорними даними

data[4] = r0.ToString();//зповнення масиву опорними даними

data[5] = "t"; //закриття блоку опорних даних

k++; //додавання лічильника

}

else

{

data[0] = "o"; //відкриття блоку даних

data[1] = db.ToString();//заповнення масиву поточними даними

data[2] = dl.ToString();//заповнення масиву поточними даними

data[3] = dt.ToString();//заповнення масиву поточними даними

data[4] = dr.ToString();//заповнення масиву поточними даними

data[5] = "c"; //закриття блоку даних

}

File.AppendAllLines(Application.StartupPath + "/Database.txt", data);

//додавання поточних даних до файлу

}

if (playing) //перевірка режиму відтворення

{

string[] reda = File.ReadAllLines(Application.StartupPath + "/Database.txt"); //зчитування даних з файлу та переведення їх до масиву

string[] curr = new string[6]; //масив поточних даних для відтворення

try

{

curr[0] = reda[k \* 6 + 0]; //заповнення поточного масиву

curr[1] = reda[k \* 6 + 1]; //заповнення поточного масиву

curr[2] = reda[k \* 6 + 2]; //заповнення поточного масиву

curr[3] = reda[k \* 6 + 3]; //заповнення поточного масиву

curr[4] = reda[k \* 6 + 4]; //заповнення поточного масиву

curr[5] = reda[k \* 6 + 5]; //заповнення поточного масиву

}

catch (Exception)

{

playing = false; //зупинка відтворення, коли закінчились дані

MessageBox.Show("Відтворення завершено");

}

if(curr[0] == "s") //перевірка, якщо поточний блок – опорний

{

b0 = int.Parse(curr[1]); //введення опорних даних із масиву

l0 = int.Parse(curr[2]); //введення опорних даних із масиву

t0 = int.Parse(curr[3]); //введення опорних даних із масиву

r0 = int.Parse(curr[4]); //введення опорних даних із масиву

}

else

{

if(curr[0] == "o") //перевірка на початок блоку поточних даних

{

db = int.Parse(curr[1]); //введення поточних даних

dl = int.Parse(curr[2]); //введення поточних даних

dt = int.Parse(curr[3]); //введення поточних даних

dr = int.Parse(curr[4]); //введення поточних даних

}

}

k++;

}}}

### ВИВІД РЕЗУЛЬТАТІВ КОРИСТУВАЧУ

Результати, що отримує користувач проекту повністю надані в графічному вигляді, для легшого зрозуміння, наглядності та загальної зручності.

Графічний інтерфейс програми виконаний на основі елемента pictureBox із застосуванням графічних можливостей C#. Реалізоване відображення графіки початково створюється за допомогою елементу bitmap. В ньому виконується початкова розмітка відносних координат, нанесення в них моделі спостережуваного об’єкта. Головною перевагою на користь bitmap в цьому випадку виступає те, що за його використання графіка виглядає більш суцільною та відсутнє неприємне явище – “блимання” графічної області.

for (int i = 54; i < 648; i += 54) //нанесення координатної сітки

{

gr.DrawLine(Pens.DarkViolet, i, 0, i, 432);

gr.DrawLine(Pens.DarkViolet, 0, i, 648, i);

}

for (int c = 215; c <= 217; c++) //нанесення вертикальної опорної лінії

{

gr.DrawLine(Pens.DarkViolet, 0, c, 648, c);

}

for (int c = 323; c <= 325; c++) //нанесення горизонтальної

{ //опорної лінії

gr.DrawLine(Pens.DarkViolet, c, 0, c, 432);

}

gr.FillRectangle(Brushes.DarkTurquoise, (284 + delaw), (176 + delah), 80, 80); //нанесення моделі досліджуваного об’єкта (площина XY)

# ПОСІБНИК КОРИСТУВАЧА

Головне меню програмного проекту складається з 5 пунктів, з яких 3 є основними, 2 – додатковими:

1. Пункт «Title» – містить інформацію про тему та авторство.
2. Пункт «Theory» – містить основні теоретичні відомості щодо принципів, якими керувався автор при створенні проекту, обґрунтування використаних методів дослідження.
3. Пункт «Mode 1» – містить програмний елемент, що графічним способом показує відстежуване в реальному часі переміщення об’єкту спостереження у площині XY.
4. Пункт «Mode 2» – містить програмний елемент, що графічним способом показує відстежуване в реальному часі переміщення об’єкту спостереження по осі Z
5. Пункт «FAQ» – містить відомості про те, як користуватися програмою та відповіді на часто проголошувані питання.

Також, над кнопками головного меню присутні дві кнопки: ліва – розпочати запис, права – відтворити запис.

При натисканні кнопки запису усі переміщення об’єкту з моменту натискання, до моменту повторного натискання, або відтворення, запишуться у файл Database.txt, за адресою: KwalWork\_Project\KwalWork\_Project\bin\Debug\.

При натисканні кнопки відтворення почнуть показуватися усі записані в цей текстовий файл дані переміщення та їх можна буде переглянути за допомогою тих самих вікон, у яких відбувається показ переміщення в реальному часі.

# СТРУКТУРНА СХЕМА

# ВИСНОВКИ

Під час виконання випускної роботи був розроблений пристрій із необхідними програмними засобами, що відповідають поставленим на початку вимогам.

Зручний та компактний прилад, створений із доступних елементів достатньо добре виконує свою задачу із збору даних що задовольняють поставленим вимогам до точності. Може бути використаний у системах спостереження як недорогий додаток, або може бути повноцінною системою, до того ж при подальшому розвитку з використанням покращеної практичної бази може бути значно піднятий клас точності, що в перспективі дозволяє отримати альтернативний варіант системи безпеки.

Програмні елементи спільно функціонують та добре розкривають потенціал приладу. Отримувані вихідні дані піддаються коректній обробці, а подальша їх візуалізація відображає те, що отримані результати несуть достатню практичну вагу за наявної бази.

Поставлена мета досягнута, задачі виконані, обґрунтовані варіанти розвитку та практична доцільність дослідження.