МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Кафедра “Підйомно-транспортного та робототехнічного обладнання"

# МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ

з дисципліни «Комп'ютерні методи розрахунку роботів»

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем,*
*Мехатроніка та промислові роботи*

ОДЕСА: ОНПУ, 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Кафедра “Підйомно-транспортного та робототехнічного обладнання"

Михайлов Євген Павлович

# МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ

з дисципліни «Комп'ютерні методи розрахунку роботів»

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем*,
*Мехатроніка та промислові роботи*

Затверджено
на засіданні кафедри
підйомно-транспортного і
робототехнічного обладнання
Протокол № від __ _____2021 р.

ОДЕСА: ОНПУ, 2021

Методичні вказівки до лабораторних занять з дисципліни «Комп'ютерні методи розрахунку роботів» для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальності: 131 - Прикладна механіка, освітні програми: – Мехатроніка та промислові роботи, Інженерія логістичних систем, / Укл.: Михайлов Є. П. – Одеса: ОНПУ, 2021. - 54 с.

Укладач:
Михайлов Є.П. доц. кафедри підйомно-транспортного і робототехнічного обладнання

**Зміст**

## Вступ

В методичних вказівках наведені лабораторні роботи з дисципліни «Комп'ютерні методи розрахунку роботів». При цьому основна увага уділялась питанням використання комп’ютерних методів розрахунку параметрів керування промисловими стаціонарними та мобільними роботами, які здійснюються самою комп’ютерною системою керування роботом.

Засоби керування роботами задають параметри переміщення робочого органу або самого робота, тому система керування повинна перерахувати ці параметри в параметри, що здійснюють керування виконавчими пристроями.

У лабораторних роботах розглянуті приклади використання комп’ютерних методів розрахунку для вирішування задач прямої та зворотної кінематики, а також обробки інформації для вирішання задач персональної та локальної навігації мобільних роботів.

Для дослідження принципів вирішування таких задач була обрана система керування на основі програмно-апаратного комплексу Arduino, програмування якого здійснюється за допомогою безкоштовного програмного забезпечення Arduino IDE (інтегрована середа розробки Arduino IDE). Програмування цієї системи здійснюється за допомогою спрощеного варіанту мову програмування C++, яка є однією з найбільш поширених мов програмування.

Дослідження засобів, що використовують комп’ютерні методи розрахунку, здійснюється на основі макетів стаціонарних та мобільних роботів, виготовлених у межах студентських науково-технічних гуртків на основі контролерів Arduino.

**Лабораторне заняття 1 Використання комп'ютерних розрахунків для визначення параметрів керування маніпулятора з циліндричною системою координат**

**Мета заняття:** Дослідження засобів визначення параметрів керування маніпулятора з циліндричною системою координат для переміщення робочого органу за допомогою комп'ютерних розрахунків, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно.

**Тривалість заняття:** 4 години.

**1.Теоретическая частина**

Розглянемо визначення параметрів керування за допомогою комп'ютерних розрахунків на прикладі маніпулятора з циліндричною системою координат, який має одну ланку та робочий орган, що переміщується вздовж ланки (рис. 1).

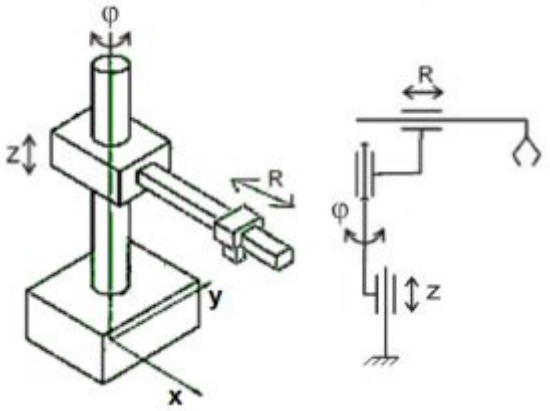



Рис. 1. Маніпулятор з циліндричною системою координат

Маніпулятор має одну ланку (рука), та робочий орган, що переміщується вздовж руки.

Переміщення руки здійснюється шляхом повороту на кут φ, а робочий орган переміщується на відстань *R*, що дає можливість встановити його в задану позицію робочої зони з координатами (x y).

Переміщення у тривимірному просторі здійснюється за рахунок підйому і опускання відносно осі z.

Тому розглянемо переміщення маніпулятора в одній площині аналогічно тому, як наведено на рис. 2.

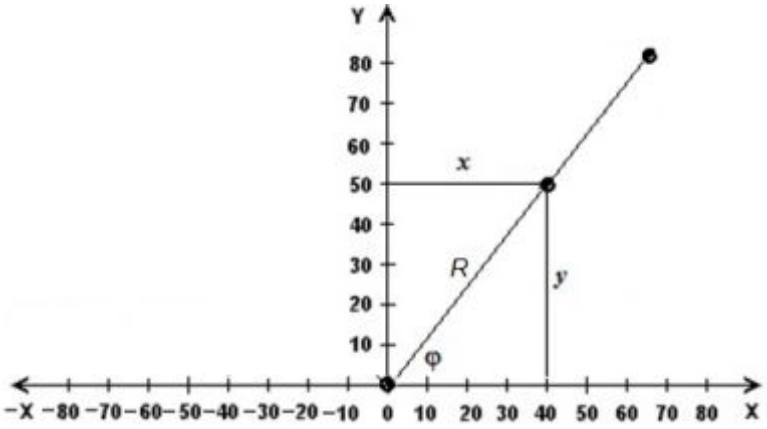



Рис. 2. Переміщення маніпулятора в одній площині

Координати робочого органу x, y визначають кут повороту ланки маніпулятора φ та положення робочого органу на ланці *R*, або навпаки, кут повороту φ та положення робочого органу на ланці *R* визначають координати робочого органу (x y).

Таким чином, керування маніпулятором здійснюється виконавчими приладами з циліндричною системою координат, а переміщення робочого органу встановлюється лінійною (прямокутною) системою координат.

Розглянемо, як здійснюється перехід від однієї системи координат до другої, а саме:

від прямокутної до циліндричної системи координат для розрахування параметрів керування виконавчими приладами *R* та φ виходячи з координат x, y,

від циліндричної до прямокутної системи координат для розрахування координат x, y виходячи з параметрів керування виконавчими приладами *R* та φ.

Оскільки параметр *z* для обох систем координат не змінюється визначено, які розрахунки треба провести для здійснення переходу від циліндричної до прямокутної системи координат та навпаки.

Виходячи з рис. 2 положення робочого органу на ланці *R* та кут повороту ланки маніпулятора φ пов'язані з координатами x, y такими залежностями:

$$R = (x^2 + y^2)^{1/2}; \quad (1)$$

$$\varphi = \operatorname{atan}(y / x). \quad (2)$$

Координати положення робочого органу *x, y, z* в тривимірному просторі пов'язані з положенням робочого органу на ланці *R* та кутом повороту ланки маніпулятора $\varphi$ такими залежностями:

$$x = R \cos \varphi; \quad (3)$$

$$y = R \sin \varphi. \quad (4)$$

У разі використання аналітичного програмування система керування роботом повинна визначити положення робочого органу на ланці *R* та кут повороту ланки маніпулятора φ, виходячи з заданих координат робочого органу x, y, використовуючи залежності (1) та (2), або навпаки, використовуючи залежності (3) та (4).

**2. Практична частина**

Визначення вказаних параметрів треба зробити за допомогою комп'ютерних розрахунків, що здійснює система керування роботом на основі апаратно-програмного комплексу Ардуіно, який використовує мову програмування C++ [2].

Створення програм для контролерів Ардуіно здійснюється за допомогою безкоштовного програмного забезпечення Arduino IDE (інтегрована середа розробки Arduino IDE), який можна завантажити з [6].

Розглянемо оператори та функції, які треба використовувати для складання програми визначення параметрів згідно з отриманими формулами.

**Складання, віднімання, множення і ділення**

Оператори +, -, * і / відповідно, повертають результат виконання арифметичних дій над двома операндами. Результат, що повертається, буде залежати від типу даних операндів.

**Квадратний корінь числа.**

sqrt(x)

Параметри:

x: число, будь-який тип даних.

Значення, що повертаються:

double, квадратний корінь числа.

**Квадрат числа.**
sq(x)
Параметри:
x: число, будь-який тип даних.
Значення, що повертаються:
квадрат числа.
**Тригонометричні функції**
sin(rad),
cos(rad),
tan(rad).
Параметри:
rad: кут у радіанах, тип даних float.
Значення, що повертаються:
тригонометрическая функція, тип даних double.
**Зворотні тригонометричні функції**
asin(x), acos(x), atan(x).
зворотна тригонометрическая функція повертає дані у радіанах, тип даних float.

**Приклад програми виконання арифметичних дій**

Створимо програму для послідовності арифметичних дій з отриманням результату у форматі float (з плаваючою комою) згідно з формулою:

$$R= ( x^2 + y^2)^{1/2}$$

```
sketch_arithmetic_20_02_25

float X; //вихідні дані X
float Y; //вихідні дані  Y
float R; //результат R

void setup() {
Serial.begin(9600);
Serial.println("R = sqrt(sq(X) + sq(Y))");
}

void loop() {
// Коли в буфері є дані
while (Serial.available() > 0)
{
X = Serial.parseFloat(); // Отримане число
Y = Serial.parseFloat(); // Отримане число
if (Serial.read() == '\n') // Кінець передачі
{
R = sqrt(sq(X) + sq(Y));
Serial.print("X = \t");
Serial.println(X, 4); //вихідні дані X
Serial.print("Y = \t");
Serial.println(Y, 4); //вихідні дані  Y
Serial.print("R = \t");
Serial.println(R, 4); //результат Z
}
}
}
```

Для обміну даними між контролером і комп'ютером використовують Монітор порту Ардуіно та Плотер по послідовному з'єднанню.

Монітор порту Ардуіно та Плотер по послідовному з'єднанню - це утиліти, які вбудовані в середу програмування Arduino IDE і служить для зв'язку комп'ютера з контролером.

Монітор порту видає дані у символьному вигляді.

Плотер по послідовному з'єднанню видає дані у графичному вигляді.

Принцип дії утиліт Монітор порту Ардуіно та Плотер по послідовному з'єднанню наведений у додатку 1.

На рис. 3 наведені дані, що були отримані на моніторі при обчисленні наведеної вище функції для значень x = 2, y = 3 (результат R =3,6056). Числа, які вводяться, розділяються комою.

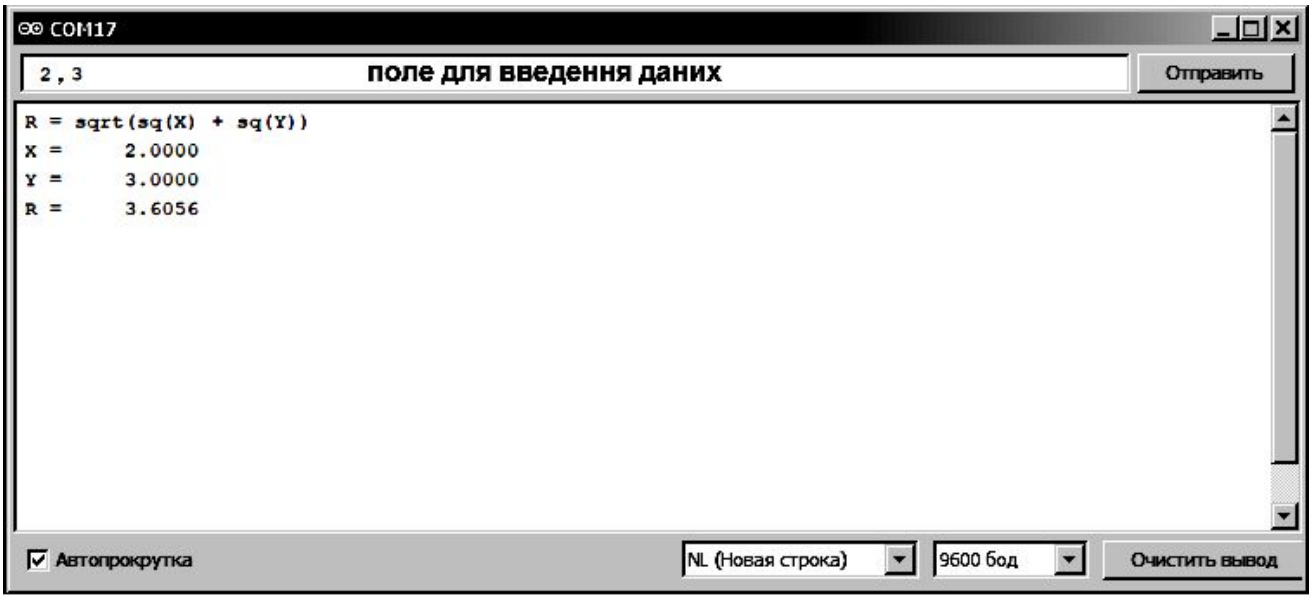



Рис. 3. Результат, отриманий на моніторі

**Приклад програми для визначення зворотних тригонометричних функцій (acos).**

Вихідний параметр задається на моніторі у форматі з плаваючою комою, оскільки він має значення від 0 до 1.

Отримаємо результат у радіанах у форматі з плаваючою комою (float), та градусах у форматі з плаваючою комою та форматі цілих чисел (int).

```
sketch_ atan_ float

float X; //вихідні дані X
float Y; //вихідні дані  Y
float PhiRad; //результат кута у радіанах
float PhiDeg; //результат кута у градусах

void setup() {
Serial.begin(9600);
Serial.println("Phi = atan (Y / X)");
}

void loop() {
// Коли в буфері є дані
while (Serial.available() > 0)
{
X = Serial.parseFloat(); // Отримане число
Y = Serial.parseFloat(); // Отримане число
```

```
if (Serial.read() == '\n') // Кінець передачі
{
PhiRad= atan (Y / X);  //результат у радіанах
PhiDeg = PhiRad * 180.0 / PI; //результат у градусах
Serial.print("X \t");
Serial.println(X, 4); //вихідні дані
Serial.print("Y \t");
Serial.println(Y, 4); //вихідні дані
Serial.print("Phi radians \t");
Serial.println(PhiRad, 4); //результат у радіанах
Serial.print("Phi degrees \t");
Serial.println(PhiDeg, 2); //результат у градусах
}
}
}
```

На рис. 4 наведені дані отримані на моніторі при обчисленні функції atan (Y/X) для значень X = 10, Y = 10.


Рис. 4. Результат, отриманий на моніторі

**Приклад розв'язання задачі**

На рис. 5 наведений результат обчислювання для значень, визначених на рис. 6:
x = 10, y = 10.
Результат, отриманий на калькуляторі, дорівнює:
$R = (x^2 + y^2)^{1/2} = (10^2 + 10^2)^{1/2} = 14{,}14213562373095.$
$\varphi$ = atan ( y / x) = arctg (10 / 10) = 45,00;

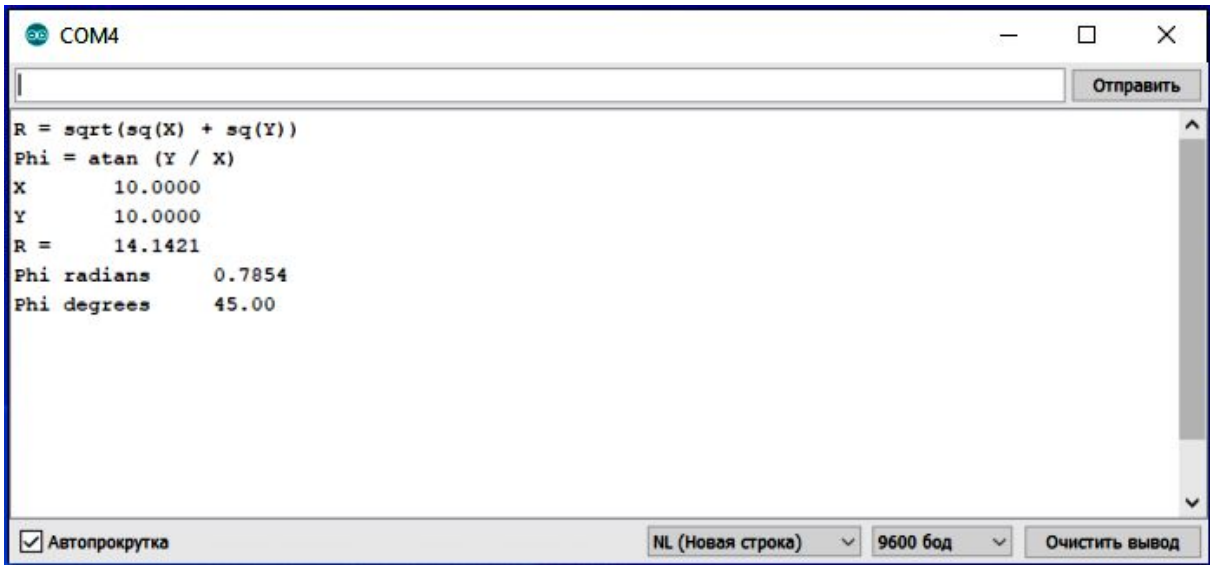


Рис. 5. Результат, отриманий на моніторі

**Приклад програми з використанням тригонометричних функцій**
Створимо програму згідно з формулою:

$$x = R \cos \varphi$$

sketch_Rcos

```
float R; //вихідні дані положенням робочого органу на ланці R
float PhiDeg; //вихідні дані кута повороту ланки маніпулятора у градусах
float PhiRad; //вихідні дані кута повороту ланки маніпулятора у радіанах
float X; //результат X

void setup() {
Serial.begin(9600);
Serial.println("X = R * cos (Phi)");
}
void loop() {
// Коли в буфері є дані
while (Serial.available() > 0)
{
R = Serial.parseFloat(); // Отримане число R
PhiDeg = Serial.parseFloat(); // Отримане число φ у градусах
if (Serial.read() == '\n') // Кінець передачі
{
PhiRad = PhiDeg * DEG_TO_RAD;
X = R * cos(PhiRad);
Serial.print("R \t");
Serial.println(R, 4); //вихідні дані R
Serial.print("Phi degrees \t");
Serial.println(PhiDeg, 2); //вихідні дані кута повороту у градусах
Serial.print("X \t");
Serial.println(X, 4); //результат X
}
}
}
```

Для перевірки програми використовується симулятор UnoArduSim.

На рис. 6 наведений результат обчислювання, отриманий за допомогою симулятора UnoArduSim для значень: R = 10, φ = 45°.

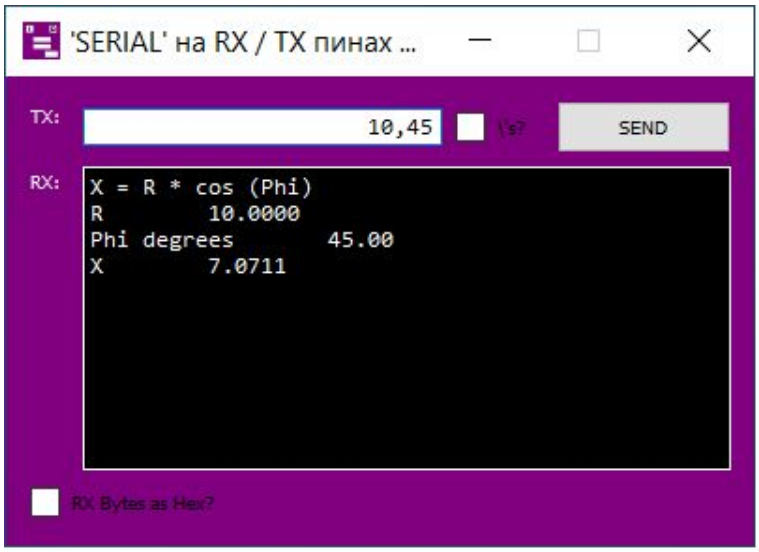



Рис. 6. Результат обчислювання, отриманий за допомогою симулятора UnoArduSim

**Завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів приводів (кут повороту та переміщення вздовж руки) для переміщення робочого органа у визначену позицію.
2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з використання різних математичних функцій.
3. Ознайомитись з програмами для обчислення проміжних даних.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення проміжних даних на стенді, використовуючи результати, що отримані у прикладі.

**Завдання 2**

1. Використовуючи приклади, скласти програму для обчислення результатів *R* та φ.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення кінцевих результатів на стенді, використовуючи результати, що отримані у прикладі.
2. Ввести вихідні дані з монітору порта та визначити кут φ та переміщення вздовж руки R, що здійснюють переміщення робочого органу в задану точку (x, y) для таких варіантів:

| Варіант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| x, мм | 100 | 110 | 90 | 105 | 100 | 110 | 90 | 105 | 110 | 90 |
| y, мм | 50 | 60 | 65 | 70 | 75 | 80 | 75 | 65 | 70 | 60 |

Положення ланок відповідно результатам обчислення показати аналогічно рис. 2.

**Завдання 3**

1. Використовуючи приклади, скласти програму для обчислення результатів *x* та *y*.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення кінцевих результатів на стенді, використовуючи результати, що отримані у прикладі.

2. Ввести вихідні дані з монітору порта та визначити кут φ та переміщення вздовж руки R, що здійснюють переміщення робочого органу в задану точку (x, y) для таких варіантів:

| Варіант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| R, мм | 100 | 110 | 90 | 105 | 100 | 110 | 90 | 105 | 110 | 90 |
| φ, у градусах | 50 | 60 | 25 | 70 | 75 | 80 | 75 | 65 | 10 | 60 |

**Контрольні питання**

1. Визначити, як знайти параметри маніпулятора з циліндричною системою координат виходячи з заданих координат робочого органу?
2. Описати, які функції потребуються для створення програми обчислення параметрів?
3. Назвати типи даних, що використовують обрані функції?
4. Визначити, як здійснити переміщення у тривимірному просторі?
5. Назвати , яку одиницю вимірювання використовують зворотні тригонометричні функції?

**ПРОТОКОЛ**
**до лабораторного заняття 1**
**Використання комп'ютерних розрахунків для визначення параметрів керування маніпулятора з циліндричною системою координат**

**Студент _____ група _____**

**Мета заняття:** Дослідження засобів визначення параметрів керування маніпулятора з циліндричною системою координат для переміщення робочого органу за допомогою комп'ютерних розрахунків, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно.

**Опис лабораторної установки**

Використовується модель маніпулятора з циліндричною системою координат та одною ланкою, обертання якої здійснюється за допомогою сервопривода з орієнтацією згідно рис. 1. Сервопривод здійснюють повертання осі в діапазоні від 0° до 180° з кроком 1°. Переміщення робочого органу вздовж руки здійснюється за допомогою крокового двигуна, що забезпечую точність позиціонування ±1мм.

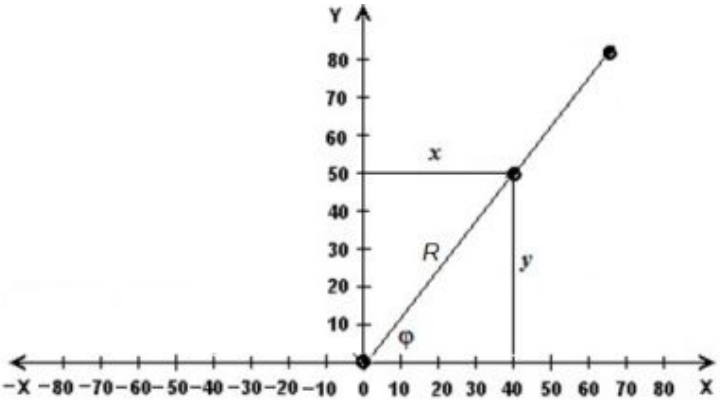


Рис. 1. Модель маніпулятора

Формули для визначення зв'язку між системами координат

$R = (x^2 + y^2)^{1/2}$;
$\varphi = \text{atan}(y / x)$.
$x = R \cos \varphi$;
$y = R \sin \varphi$.

**Домашнє завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів приводів (кут повороту та переміщення робочого органу вздовж руки) для переміщення у визначену позицію.
2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з
3. Скласти програми для обчислення проміжних даних.

**Домашнє завдання 2**

1. Скласти програму для обчислення кінцевих результатів.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення проміжних даних та кінцевих результатів на стенді, використовуючи результати, що отримані у прикладі.
2. Визначити кут φ та переміщення робочого органу вздовж руки R, що здійснюють переміщення робочого органу в задану точку (x, y) для таких вихідних даних:

| Вихідні дані: | Знаходимо: | Результат, отриманий на калькуляторі: |
|---|---|---|
| x = ____ мм, | φ = _____°, | φ = _____°, |
| y = ____ мм, | R = _____ мм. | R = _____ мм. |

3. Визначити кут φ та переміщення робочого органу вздовж руки R, що здійснюють переміщення робочого органу в задану точку (x, y) для таких вихідних даних:

| Вихідні дані: | Знаходимо: | Результат, отриманий на калькуляторі: |
|---|---|---|
| φ = ____ °, | x = _____ мм, | x = _____ мм, |
| R = ____ мм, | y = _____ мм. | y = _____ мм. |

Порівняти дані, отримані на калькуляторі та контролері.

**Контрольні питання**

1. Визначити, як знайти параметри маніпулятора з циліндричною системою координат виходячи з заданих координат робочого органу ?

_____
_____

2. Описати, які функції потребуються для створення програми обчислення параметрів?

_____
_____

3. Назвати типи даних, що використовують обрані функції?

_____
_____

4. Визначити, як здійснити переміщення у тривимірному просторі?

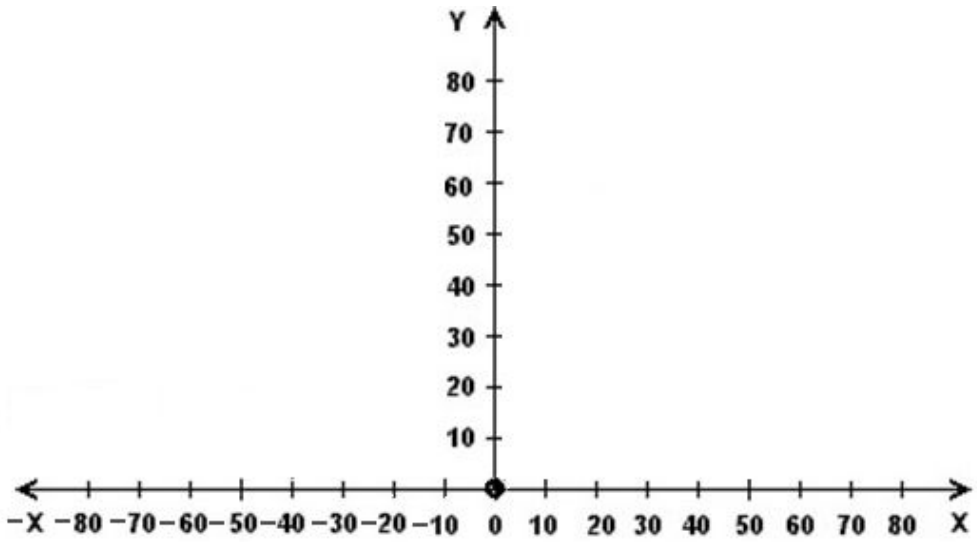
_____
_____

5. Назвати, яку одиницю вимірювання використовують зворотні тригонометричні функції?

_____
_____

**Результати виконання завдання**

Положення ланок відповідно результатам обчислення показати на рисунку.




**Висновки**

_____
_____
_____
_____

_____

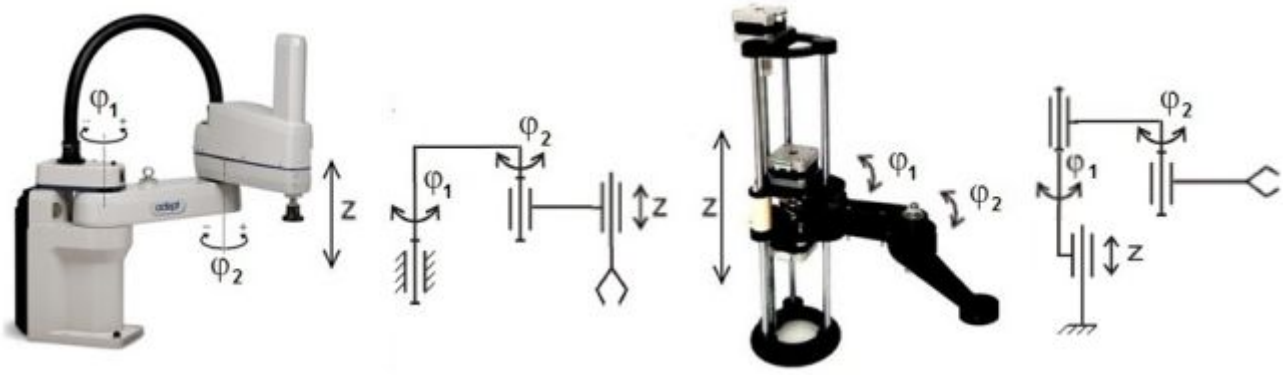
**Лабораторне заняття 2 Використання комп'ютерних розрахунків для визначення параметрів керування маніпулятора SCARA**

**Мета заняття:** Дослідження засобів визначення параметрів керування маніпулятора SCARA для переміщення робочого органу за допомогою комп'ютерних розрахунків на основі зворотної задачі кінематики, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно.

**Тривалість заняття:** 2 години.

**1.Теоретическая частина**

Розглянемо визначення параметрів керування за допомогою комп'ютерних розрахунків на прикладі маніпулятора SCARA, що здійснює операції підйому та повороту з зміною відстані та орієнтації початкового та кінцевого положення захоплюючого пристрою (рис. 1).




а) б)

Рис. 1. Маніпулятор SCARA

Маніпулятор має дві ланки (плече і рука), що здатні працювати тільки в одній площині.

Переміщення ланок здійснюється шляхом повороту на кути $\varphi_1$ та $\varphi_2$.

Переміщення у тривимірному просторі здійснюється за рахунок підйому і опускання відносно осі z захоплюючого пристрою (рис. 1,а) або руки (рис. 1,б).

Тому розглянемо переміщення маніпулятора в одній площині аналогічно тому, як наведено у [1] (рис. 2).

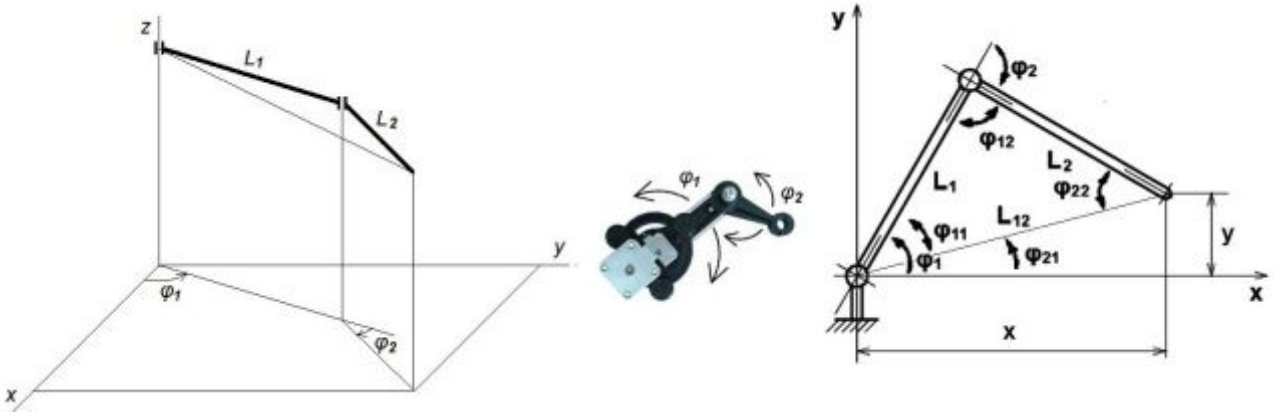



Рис. 2. Переміщення маніпулятора в одній площині

Визначення кутів $\varphi_1$ і $\varphi_2$, які дозволять маніпулятору із ланками $L_1$ і $L_2$ помістити робочий орган в задану точку (x, y), можна зробити за допомогою зворотної задачі кінематики.

Проведемо пряму $L_{12}$, що сполучає початок координат O із заданою точкою (x, y).

$$L_{12}^2 = x^2 + y^2,$$

звідси маємо

$$L_{12} = ( x^2 + y^2)^{\frac{1}{2}}.$$

На попередньому рис. 2 видно, що

$$\varphi_1 = \varphi_{11} + \varphi_{21}.$$

де

$\varphi_{11}$ - кут між прямими $L_1$ та $L_{12}$,

$\varphi_{21}$ - кут між віссю 0X і прямий $L_{12}$.

Кут $\varphi_{21}$ знаходимо, виходячи з того, що

$$x = L_{12}\cdot\cos(\varphi_{21}), \text{ або}$$
$$y = L_{12}\cdot\sin(\varphi_{21}),$$

Звідси маємо:

$$\varphi_{21} = \arccos( x/ L_{12} ) \text{ або } \varphi_{21} = \operatorname{arctg}( y/x ).$$

Кут $\varphi_{11}$ знаходимо за допомогою теореми косинусів, згідно з якою для плоского трикутника зі сторонами a, b, c і кутом $\alpha$, протилежним стороні a, справедливо співвідношення:

$$a^2 = b^2 + c^2 - 2\cdot b\cdot c\cdot\cos(\alpha).$$

За теоремою косинусів маємо:

$$L_2^2 = L_{12}^2 + L_1^2 - 2\cdot L_{12}\cdot L_1\cdot\cos(\varphi_{11}),$$

звідси отримаємо

$$\varphi_{11} = \arccos( ( L_1^2 - L_2^2 + L_{12}^2 ) / 2\cdot L_{12}\cdot L_1 ),$$
$$\varphi_1 = \varphi_{11} + \varphi_{21} = \arccos( x/ L_{12} ) + \arccos( ( L_1^2 - L_2^2 + L_{12}^2 ) / 2\cdot L_1 \cdot L_{12}).$$

Знайдемо кут $\varphi_2$.

Як видно із попереднього рисунку, кут $\varphi_2$ дорівнює

$$\varphi_2 = \pi - \varphi_{12}.$$

Згідно тій же теоремі косинусів маємо

$$L_{12}^2 = L_1^2 + L_2^2 - 2\cdot L_1\cdot L_2\cdot\cos(\pi - \varphi_2),$$
$$\varphi_2 = \pi - \arccos( (L_1^2 + L_2^2 - L_{12}^2 ) / 2\cdot L_1\cdot L_2 ).$$

Таким чином були отримані залежності, за допомогою яких можна визначити зв'язок між параметрами маніпулятора, а саме, довжиною ланок $L_1$, $L_2$, кутами повороту ланок $\varphi_1$, $\varphi_2$ та положенням захоплюючого пристрою ($x_1$, $y_1$,).

Обрана кінематична схема дозволяє спростити цей зв'язок, оскільки траєкторія переміщення здійснюється в площині (x, y), а по осі z виконується тільки підйом та опускання.

Виходячи з того, що $L_{12}^2 = x^2 + y^2$, остаточно маємо

$\varphi_1 = \arccos( x / ( x^2 + y^2)^{1/2}) + \arccos( ( L_1^2 - L_2^2 + (x^2 + y^2)) / (2\cdot L_1 \cdot ( x^2 + y^2)^{1/2}))$;

$\varphi_2 = \pi - \arccos( (L_1^2 + L_2^2 – ( x^2 + y^2)) / (2\cdot L_1\cdot L_2 ))$.

Отримані залежності дають можливість знайти кути повороту ланок $\varphi_1$, $\varphi_2$ при заданих значеннях x, y.

**2. Практична частина**

Визначення вказаних параметрів треба зробити за допомогою комп’ютерних розрахунків, що здійснює система керування роботом на основі апаратно-програмного комплексу Ардуіно, аналогічно тому, як це було зроблено у попередній роботі.

**Приклад розв’язання задачі**

Для маніпулятора, що наведений на рис. 2 знайти такі кути $\varphi_1$ і $\varphi_2$, які дозволять маніпулятору із плечем $L_1$ і ліктем $L_2$ помістити робочий орган в задану точку (x, y).

Вихідні дані: $L_1 = 100$ мм, $L_2 = 100$ мм, x = 100 мм, y = 50 мм,

Знаходимо: $L_{12} = (x^2 + y^2)^{1/2} = (100^2 + 50^2)^{1/2} = 112$ мм.

Отримаємо значення кутів в радіанах:

$\varphi_1 = \arccos(x / L_{12}) + \arccos((L_1^2 - L_2^2 + L_{12}^2) / 2 \cdot L_{12} \cdot L_1) = \arccos 0{,}893 + \arccos 0{,}56 = 0{,}4668 + 0{,}9764 = 1{,}4432.$

$\varphi_2 = \pi - \arccos((L_1^2 + L_2^2 - L_{12}^2) / 2 \cdot L_1 \cdot L_2) = 3{,}14 - \arccos 0{,}375 = 3{,}1416 - 1{,}1864 = 1{,}9552$ .

Або в градусах

$\varphi_1 = 82°$, $\varphi_2 = 112$ °.

На рис. 3 наведений результат, отриманий на контролері за допомогою програми, наведеної у додатку 2.

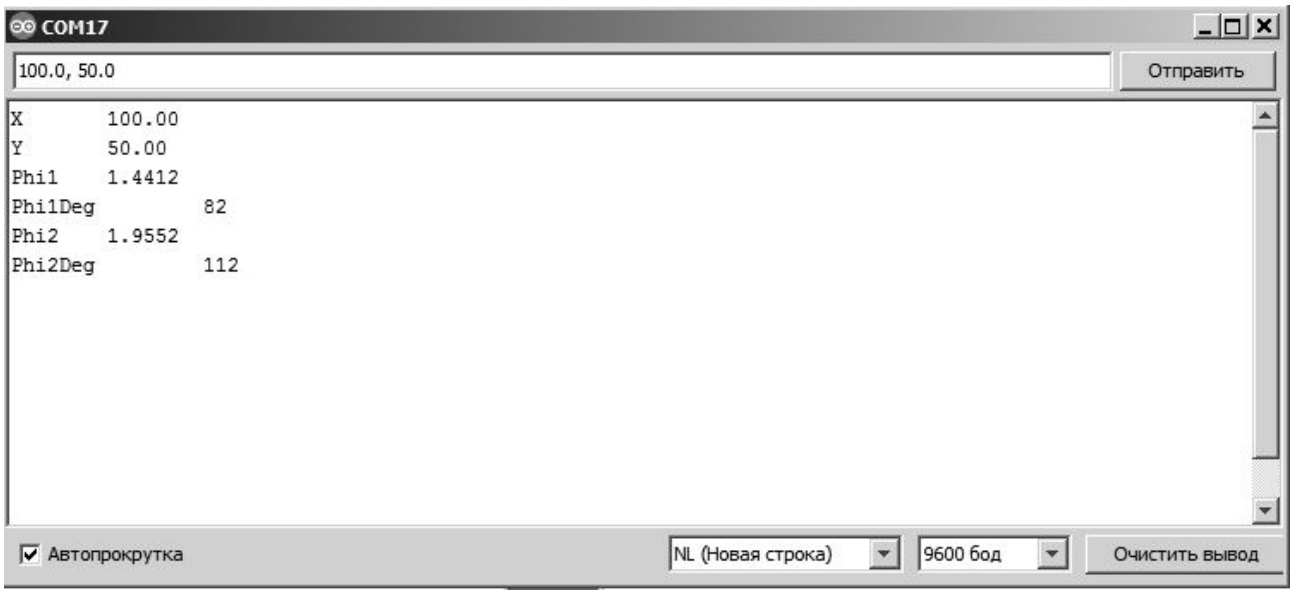



Рис. 3. Результат, отриманий на контролері

На рис. 8 наведений результат обчислень. Зміщення результату від заданої позиції визначається помилкою округлення.

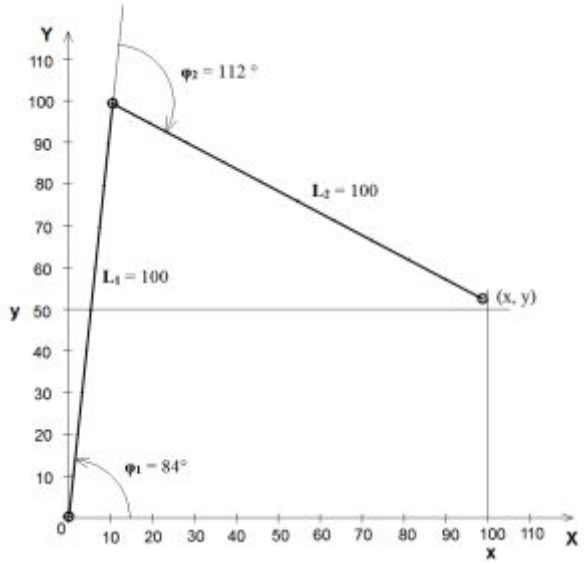



Рис. 8. Результат обчислень

**Завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів приводів (кутів повороту) для переміщення у визначену позицію.

2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з використання різних математичних функцій.
3. Скласти програми для обчислення проміжних даних.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення проміжних даних на стенді, використовуючи результати, що отримані у прикладі.

**Завдання 2**

1. Скласти програму для обчислення кінцевих результатів.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення кінцевих результатів на стенді, використовуючи результати, що отримані у прикладі.
2. Ввести вихідні дані з монітору порта та визначити кути $\varphi_1$ і $\varphi_2$ для маніпулятора із плечем $L_1$ і рукою $L_2$ , що здійснюють переміщення робочого органу в задану точку (x, y) для таких варіантів:

| Варіант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $L_1$ , мм | 60 | 80 | 90 | 65 | 75 | 85 | 95 | 105 | 110 | 90 |
| $L_2$ , мм | 55 | 75 | 80 | 65 | 70 | 85 | 90 | 100 | 100 | 80 |
| x, мм | 100 | 110 | 90 | 105 | 100 | 110 | 90 | 105 | 110 | 90 |
| y, мм | 50 | 60 | 65 | 70 | 75 | 80 | 75 | 65 | 70 | 60 |

Порівняти дані, отримані на калькуляторі та контролері.

Положення ланок відповідно результатам обчислення показати аналогічно рис. 8.

**Контрольні питання**

1. Визначити, що вирішується за допомогою зворотної задачі кінематики?
2. Описати, що таке теорема косинусів?
3. Визначити, скільки рішень має зворотна задача маніпулятора з двома ланками?
4. Визначити, який тип даних використовують зворотні тригонометричні функції?
5. Назвати, яку одиницю вимірювання використовують зворотні тригонометричні функції?

**ПРОТОКОЛ**
**до лабораторного заняття 2**
**Використання комп'ютерних розрахунків для визначення параметрів керування маніпулятора SCARA**

**Студент _____ група _____**

**Мета заняття:** Дослідження засобів визначення параметрів керування маніпулятора SCARA для переміщення робочого органу за допомогою комп'ютерних розрахунків на основі зворотної задачі кінематики, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно

**Опис лабораторної установки**

Використовується модель маніпулятора з двома ланками, обертання яких здійснюється за допомогою сервоприводів з орієнтацією згідно рис. 1. Сервоприводи здійснюють повертання осі в діапазоні від 0° до 180° з кроком 1°.

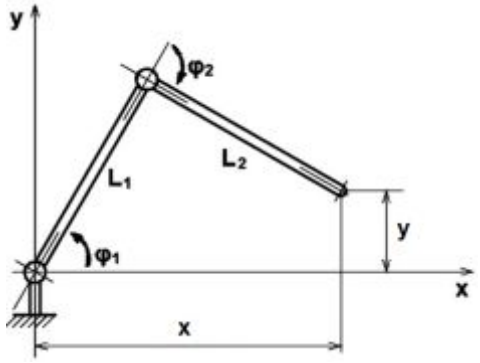



Рис. 1. Модель маніпулятора

**Домашнє завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів приводів (кутів повороту) для переміщення у визначену позицію.
2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з
3. Скласти програми для обчислення проміжних даних.

**Домашнє завдання 2**

1. Скласти програму для обчислення кінцевих результатів.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення проміжних даних та кінцевих результатів на стенді, використовуючи результати, що отримані у прикладі.
2. Визначити кути $\varphi_1$ і $\varphi_2$ для маніпулятора із плечем $L_1$ і рукою $L_2$ , що здійснюють переміщення робочого органу в задану точку (x, y) для таких вихідних даних:

Вихідні дані:
$L_1$ = _____ мм,
$L_2$ = _____ мм,
x = _____ мм,
y = _____ мм,

Знаходимо:
$\varphi_1$ = _____°,
$\varphi_2$ = _____ °.

Порівняти дані, отримані на калькуляторі та контролері.

**Контрольні питання**

1. Визначити, що вирішується за допомогою зворотної задачі кінематики?

2. Описати, що таке теорема косинусів?

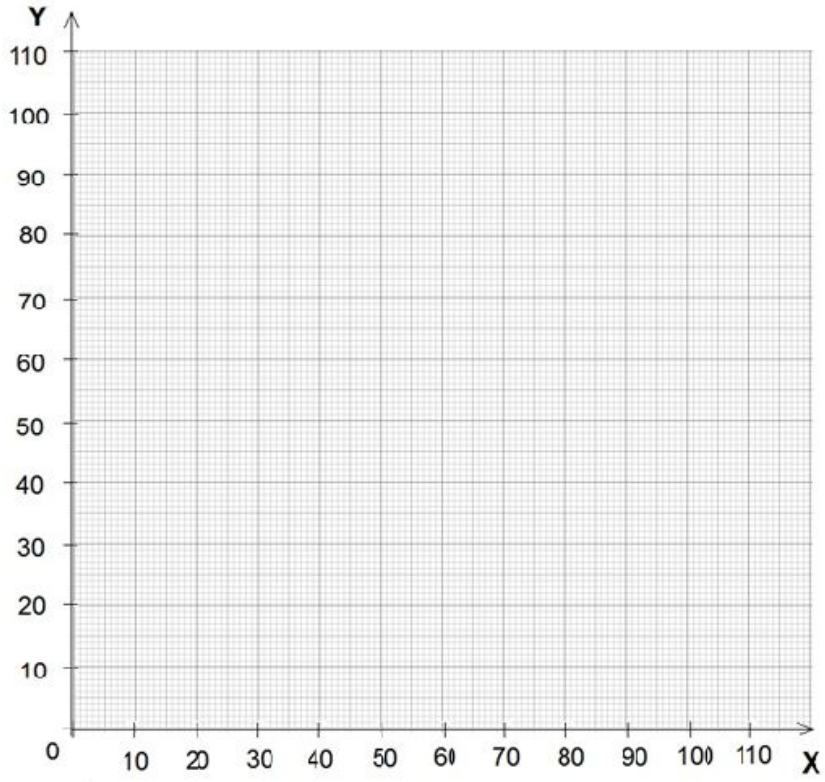
3. Визначити, скільки рішень має зворотна задача маніпулятора з двома ланками?

4. Визначити, який тип даних використовують зворотні тригонометричні функції?

5. Назвати, яку одиницю вимірювання використовують зворотні тригонометричні функції?

**Результати виконання завдання**

Положення ланок відповідно результатам обчислення показати на рисунку.




**Висновки**

**Лабораторне заняття 3 Дослідження можливості використання комп'ютерних розрахунків для орієнтації транспортного робота за допомогою лазерного сканера**

**Мета заняття:** Дослідження можливості використання комп'ютерних розрахунків для орієнтації транспортного робота за допомогою лазерного сканера, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно.

**Тривалість заняття:** 2 години

**1.Теоретическая частина**

Визначення положення та орієнтації робота можна здійснити за допомогою лазерного сканера з трьома або двома рефлекторами [3]. Такий засіб орієнтації використовують, наприклад, транспортні роботи фірми Rocla (Rocla AGV).

Так при переміщенні робота вздовж стелажу на складі у разі встановлення рефлекторів на самому стелажу є можливість обмежитись тільки двома рефлекторами, оскільки робот може знаходитись тільки по одну сторону стелажу.

Приклад використання лазерних сканерів для визначення положення робота при переміщенні вздовж стелажу наведений на рис. 1. Лазерний сканер видає відстані до рефлекторів $L_1$ та $L_2$, а також відповідні кути повороту сканера відносно орієнтації робота та рефлекторів $\alpha_1$, $\alpha_2$.

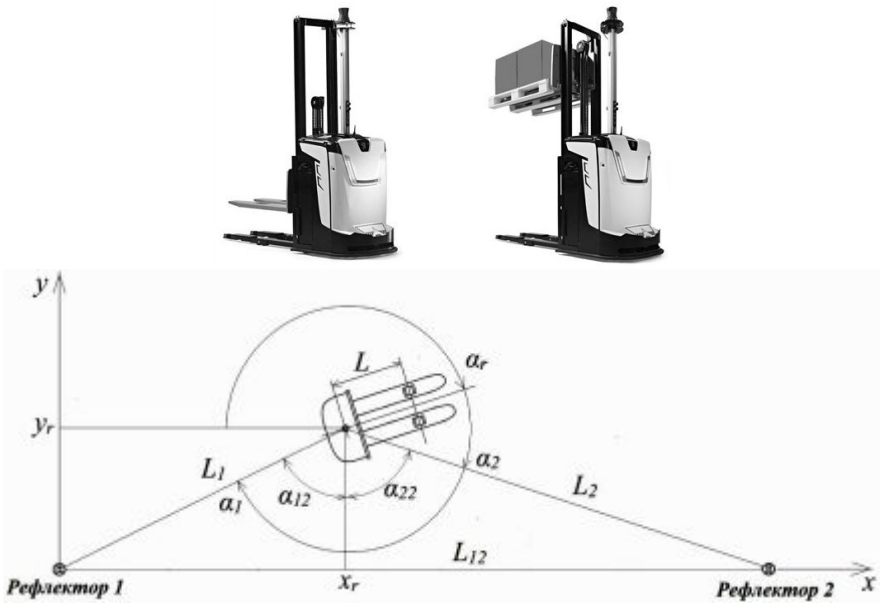



Рис. 1. Використання лазерних датчиків для визначення положення транспортного робота

На рисунку використовуються такі позначення:

$L_1$ - відстань до рефлектора 1;

$L_2$ - відстань до рефлектора 2;

$L_{12}$ – відстань між рефлекторами 1, 2;

$\alpha_1$ – кут напрямку на рефлектор 1 відносно вихідної орієнтації робота;

$\alpha_2$ - кут напрямку на рефлектор 2 відносно вихідної орієнтації робота;

$\alpha_{12}$ - кут між лінією напрямку на рефлектор 1 та лінією, перпендикулярною до лінії відстані між рефлекторами;

$\alpha_{22}$ - кут між лінією напрямку на рефлектор 2 та лінією, перпендикулярною до лінії відстані між рефлекторами;

$\alpha_r$ – орієнтація робота;

$x_r$ та $y_r$ - поточні координати робота.

Для визначення положення та орієнтації робота треба знайти його координати $x_r$ та $y_r$, а також його орієнтацію $\alpha_r$.

Вибираємо систему координат, де ось $x$ збігається з лінією відстані між рефлекторами а ось $y$ в напрямку, де здійснюється переміщення робота. Тоді $x_r$ та $y_r$ позначають поточні координати робота, що пов'язані з відстанями до рефлекторів $L_1$, $L_2$ та відстанню між рефлекторами $L_{12}$ такими залежностями:

$$y_r^2 + x_r^2 = L_1^2, \quad y_r^2 + (L_{12} - x_r)^2 = L_2^2, \quad y_r^2 = L_1^2 - x_r^2. \tag{1}$$

Після проведення наступних перетворювань:

$$L_1^2 - x_r^2 + (L_{12} - x_r)^2 = L_2^2;$$
$$L_1^2 - x_r^2 + L_{12}^2 - 2 L_{12} x_r + x_r^2 = L_2^2; \tag{2}$$
$$L_1^2 + L_{12}^2 - L_2^2 = 2 L_{12} x_r,$$

отримаємо, що $x_r$ та $y_r$ дорівнюють:

$$x_r = (L_1^2 + L_{12}^2 - L_2^2) / 2 L_{12};$$
$$y_r = (L_1^2 - x_r^2)^{1/2}. \tag{3}$$

Оскільки:

$$\cos \alpha_{22} = y_r / L_2,$$
$$\alpha_{22} = \text{arc } \cos (y_r / L_2). \tag{4}$$

Для орієнтації робота $\alpha_r$ отримаємо:

$$\alpha_r = 270° - \alpha_{22} - \alpha_2 = 270° - \alpha_2 - \text{arc } \cos (y_r / L_2). \tag{5}$$

**Приклад розв'язання задачі**

При відомому значенні $L_{12}$ =30 м, лазерний датчик визначив такі параметри:
$L_1$ = 20 м; $L_2$ = 16 м; $\alpha_1$ = 293°; $\alpha_2$ = 180°.

Для X та Y отримаємо:

$x_r = (L_1^2 + L_{12}^2 - L_2^2) / 2 L_{12} = (20^2 + 30^2 - 16^2) / 60 = (400 + 900 – 256) / 60 = 17{,}4$ м,
$y_r = (L_1^2 - x_r^2)^{1/2} = (20^2 – 17{,}4^2)^{1/2} = (400 – 302{,}76)^{1/2} = 9{,}86$ м,
$\alpha_{22} = \cos^{-1} (Y / L_2) = \cos^{-1} (9{,}86 / 16) = \cos^{-1} (9{,}86 / 16) = \cos^{-1} (0{,}61625) = 51{,}96$ °,
$\alpha_r = 270° - \alpha_{22} - \alpha_2 = 270° - 51{,}96° - 180° = 38{,}04°$.

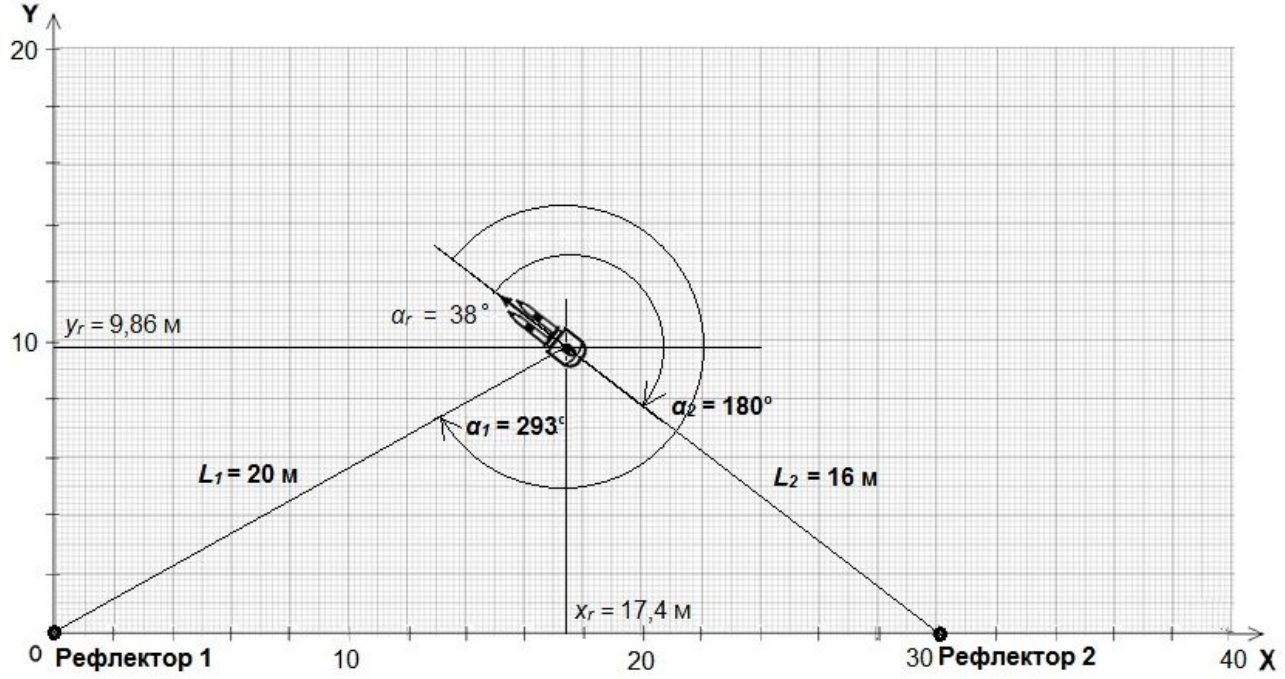



Рис. 2. Результати вимірювання вихідних даних на стенді та розрахунку

На рис. 3 наведений результат, отриманий за допомогою прикладу програми, що здійснює обчислення параметрів.

Приклад програми наведений у додатку 3.

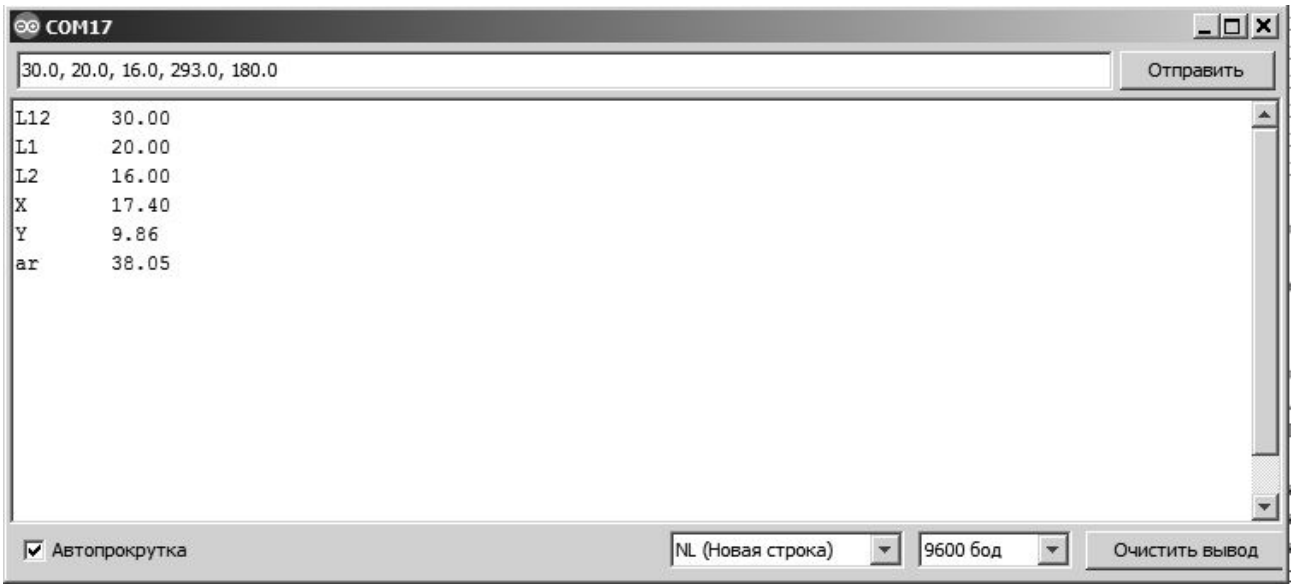



Рис. 3. Результати розрахунку, отримані на контролері

**2. Практична частина**

Визначення вказаних параметрів треба зробити за допомогою комп'ютерних розрахунків, що здійснює система керування роботом на основі апаратно-програмного комплексу Ардуіно, аналогічно тому, як це було зроблено у попередній роботі.

Порівняти дані, отримані на калькуляторі та контролері.

**Завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом Визначення положення та орієнтації робота.
2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з використання різних математичних функцій.
3. Скласти програми для обчислення кінцевих результатів.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення кінцевих результатів, а саме, координат транспортного робота $x_r$ та $y_r$, а також його орієнтацію $\alpha_r$. , використовуючи результати $\alpha_1$, $\alpha_2$, $L_1$, $L_2$, що отримані шляхом вимірювання на стенді як це показано на рис. 2.

2. Ввести вихідні дані з монітору порта та визначити координати $x_r$ та $y_r$, а також кут орієнтації $\alpha_r$. та порівняти їх з параметрами, що отримані на стенді.

**Контрольні питання**

1. Визначити, які засоби дозволяють визначити положення і орієнтацію транспортних роботів?
2. Описати, як визначити положення та орієнтацію робота за допомогою лазерного сканера з двома рефлекторами?
3. Описати, які функції потребуються для створення програми обчислення параметрів?

**ПРОТОКОЛ**
**до лабораторнго заняття 3**
**Дослідження можливості використання комп'ютерних розрахунків для орієнтації транспортного робота за допомогою лазерного сканера**

**Студент _____ група _____**

**Мета заняття:** Дослідження можливості використання комп'ютерних розрахунків для орієнтації транспортного робота за допомогою лазерного сканера, що здійснюються системою керування роботом на основі апаратно-програмного комплексу Ардуіно.

**Опис лабораторної установки**

Для визначення вказаних параметрів використовується графічний стенд. як це показано на рис. 1.

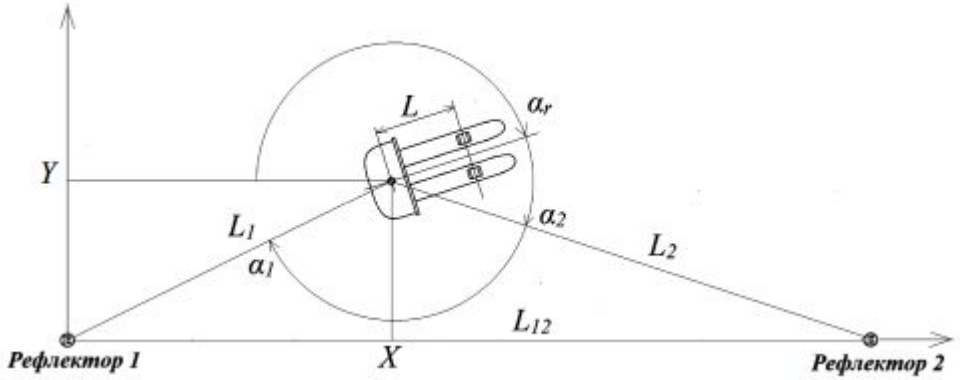



Рис. 1. Графічний стенд

**Домашнє завдання**

1. Розглянути теоретичні положення, а саме, з алгоритмом визначення положення та орієнтації робота.
2. Розкласти кінцеві формули на окремі формули обчислення проміжних даних з використання різних математичних функцій.
3. Скласти програми для обчислення кінцевих результатів.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення кінцевих результатів, використовуючи результати $\alpha_1$, $\alpha_2$, $L_1$, $L_2$, що отримані шляхом вимірювання на стенді як це показано на рис. 2.
3. Ввести вихідні дані з монітору порта та визначити координати $x_r$ та $y_r$, а також кут орієнтації $\alpha_r$. Обчислені дані порівняти з параметрами, що отримані на стенді.

**Домашнє завдання 1**

Вихідні дані:
$L_{12}$ = _____ м,
$L_1$ = _____ м,
$L_2$ = _____ м,
$\alpha_1$ = _____ °,
$\alpha_2$ = _____ °,

Знаходимо:
$x_r$ = _____ м,
$y_r$ = _____ м,
$\alpha_r$ = _____ °.

**Контрольні питання**

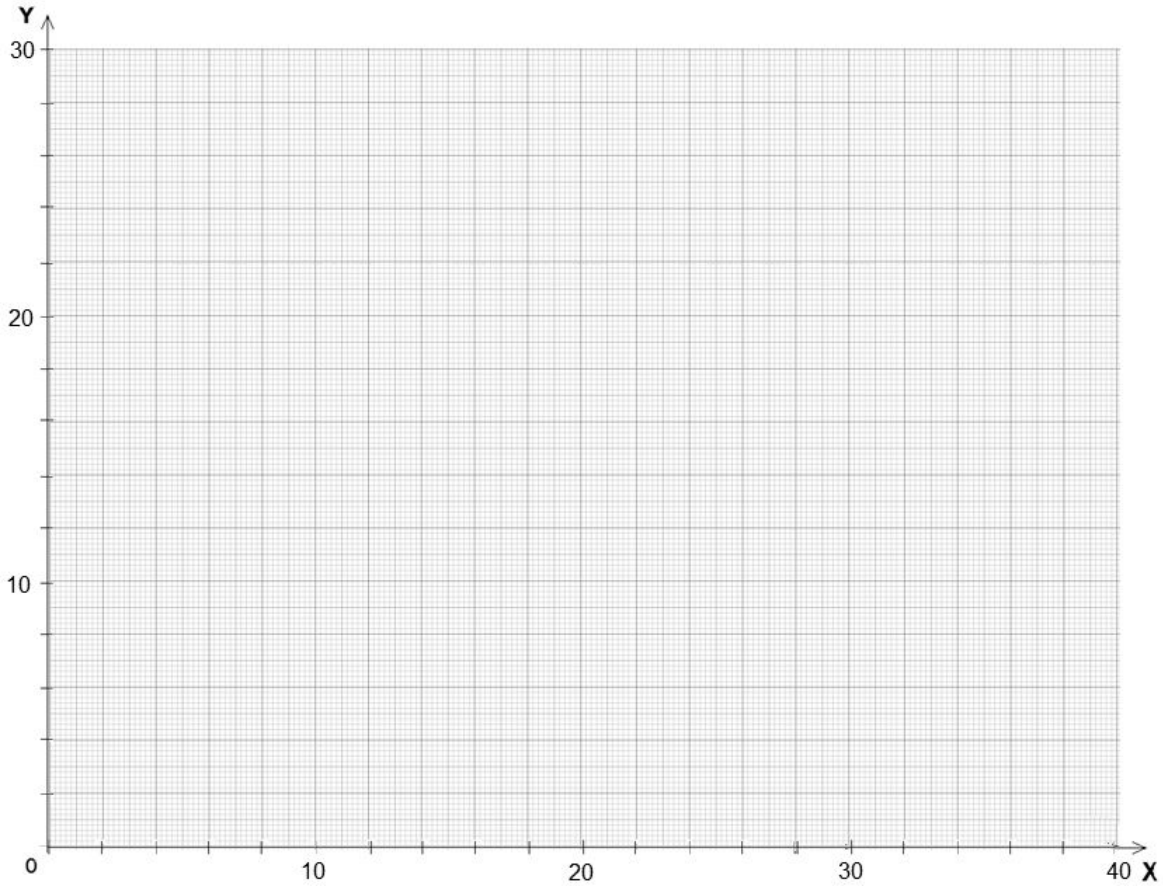
1. Визначити, які засоби дозволяють визначити положення і орієнтацію транспортних роботів?

2. Описати, як визначити положення та орієнтацію робота за допомогою лазерного сканера з двома рефлекторами?

3. Описати, які функції потребуються для створення програми обчислення параметрів?

**Результати виконання завдання**

Координати $x_r$ та $y_r$, а також кут орієнтації $\alpha_r$, що отримані у результаті обчислення.




**Висновки**

**Лабораторне заняття 4 Дослідження можливості використання комп'ютерних розрахунків для дослідження робота, який балансує**

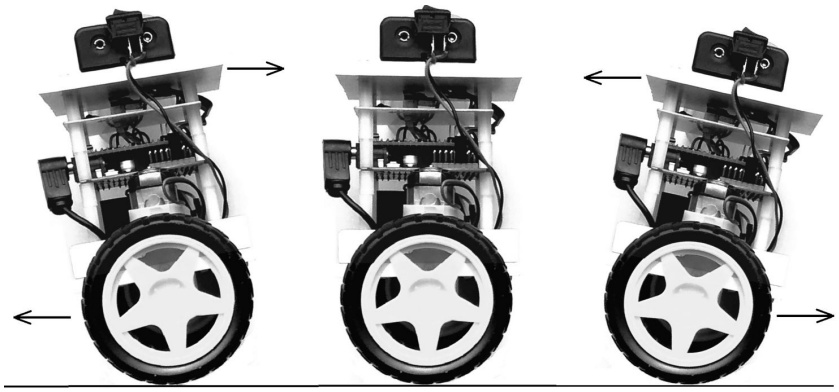
**Мета заняття:** Дослідження можливості використання комп'ютерних розрахунків для балансування двоколісного мобільного робота за допомогою гіроскопа-акселерометра MPU6050

**Тривалість заняття:** 4 години

**1.Теоретіческая частина**

У роботі розглядається икористання комп'ютерних розрахунків для створення робота, яки балансує як Сігвей (англ. Segway – двоколісний засіб для пересування стоячи, оснащений електроприводами) [6-7].

Щоб збалансувати робота, двигуни повинні протидіяти падінню робота [6] (рис. 1). Ця дія вимагає зворотного зв'язку і коригувальних елементів. Елемент зворотного зв'язку - гіроскоп-акселерометр MPU6050, який забезпечує вимірювання як прискорення, так і обертання у всіх трьох осях. Ардуіно використовує це, щоб знати поточну орієнтацію робота. Коригувальним елементом є приводи переміщення, які складаються з електродвигунів, редукторів і коліс.



а) б) в)

Рис. 1. Сбалансований (б) та нахилений (а, в) робот

Алгоритм роботи робота, яки балансує, полягає у виконанні наступній послідовності дій. Дані з гіроскопа і акселерометра у вигляді окремих компонент послідовно виходять по шині I2C, і надходять на вхід відповідних фільтрів, які згладжують шуми. При цьому використовують комплементарний фільтр, фільтр Калмана тощо. На виході фільтра отримуємо дані про становище в просторі, усереднені і очищені від шуму. У даному випадку використовується кут тільки в одній площині. Поточне положення кута передається на PID-регулятор (пропорційно-інтегрально-диференціальний), який на підставі поточного кута нахилу і його динаміки в часі приймає рішення про параметри сигналів широтно-імпульсної модуляції (прогальності, яка дорівнює відношенню періоду повторення імпульсу до його тривалості, та полярності імпульсів), що подаються на електромотори.

Принцип дії акселерометра MPU6050 наведений у додатку 4.

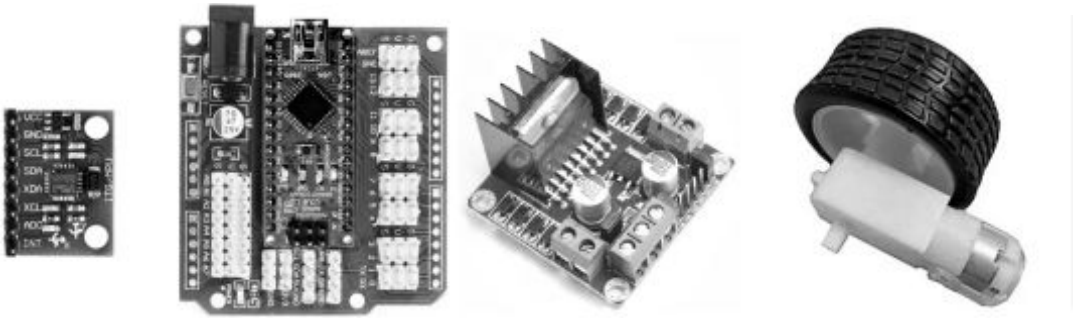
Алгоритм роботи фільтрів, які згладжують шуми, описаний у додатку 5.

Алгоритм роботи PID-регулятора описаний у додатку 6.

**2. Апаратна та програмна частина**

У склад апаратної частини входять акселерометр MPU6050, мікроконтролер Arduino Nano, модуль драйвера двигуна L298N, мотор-редуктор постійного струму з колесом (рис. 2).

.



а) б) в) г)

Рис. 2. Акселерометр MPU6050 (а), мікроконтролер Arduino Nano (б), модуль драйвера двигуна L298N (в) та мотор-редуктор постійного струму з колесом (г)

Зовнішній вигляд балансуючого робота наведений на рис. 3.

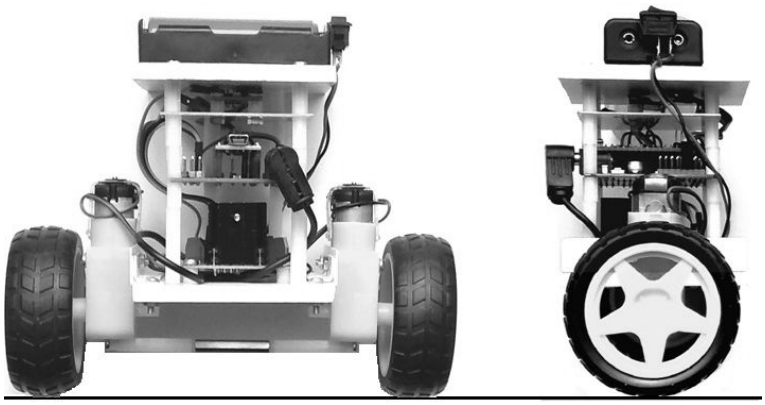


Рис. 3. Зовнішній вигляд балансуючого робота

Схема підключення датчика та приводів наведена на рис. 4.

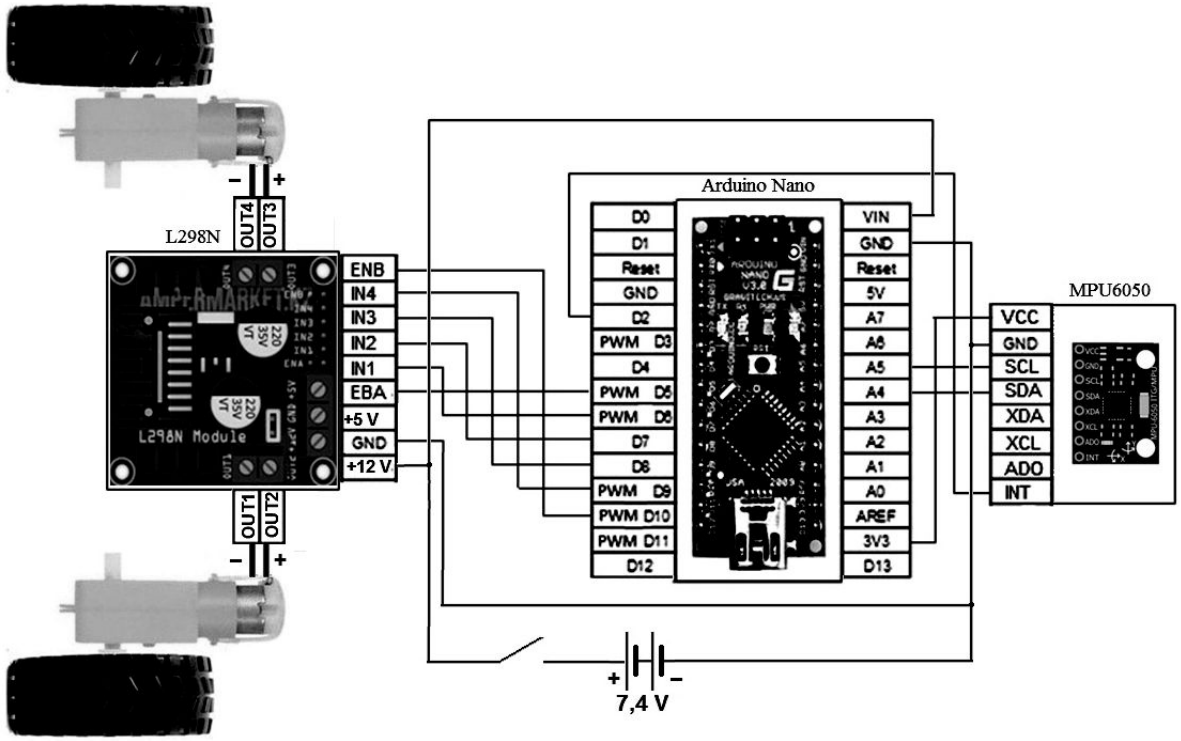



Рис. 4.Схема підключення датчика та приводів

Підключення модуля драйвера L298N:

| | | | | | | |
|---|---|---|---|---|---|---|
| L298N | ENA | IN1 | IN2 | IN3 | IN4 | ENB |
| Arduino Nano | 5 | 6 | 7 | 8 | 9 | 10 |

Підключення MPU6050 до Arduino Nano:

| | | | | | |
|---|---|---|---|---|---|
| MPU6050 | GND | VCC | SDA | SCL | INT |
| Arduino Nano | GND | +3,5V | A4 | A5 | 2 |

**3. Практична частина**

Практична частина складається з двох частин. В першій частині здійснюється дослідження датчика гіроскоп-акселерометр MPU6050. У другій частині проводиться дослідження двоколісного мобільного робота, що балансує.

**3.1. Дослідження датчика гіроскоп-акселерометр MPU6050**

Программа дослідження датчика складається з послідовності таких дій.

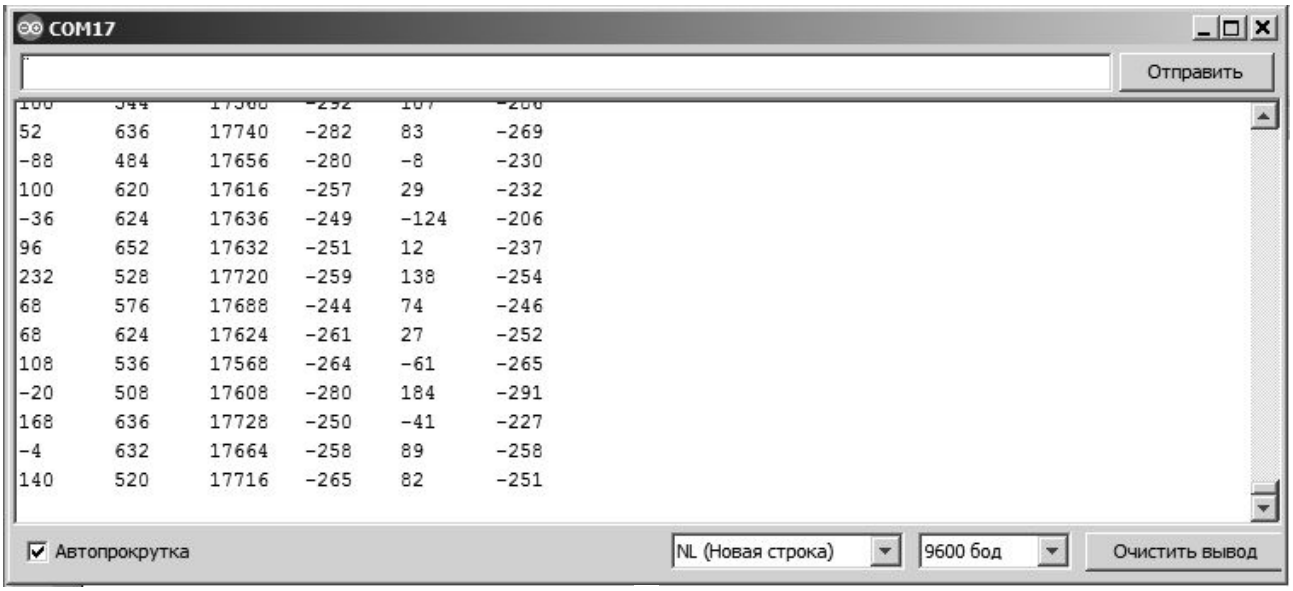
Спочатку створюється програма, що здійснює вивід на монітор даних датчика MPU6050.

У програмі використовуються бібліотеки для датчика MPU6050, яка дозволяє визначити усі параметри, та. бібліотека Wire, яка призначена для зв'язку I2C між датчиком Arduino і MPU6050. Тому спочатку треба підключити ці бібліотеки.

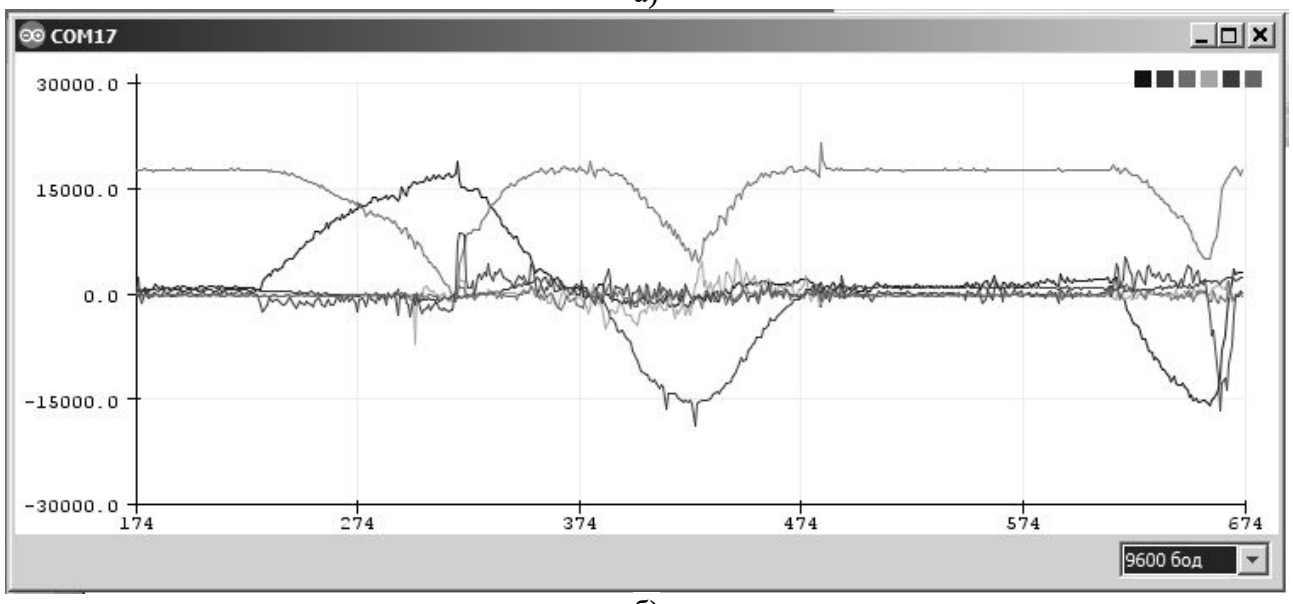
```
#include <Wire.h>
#include <MPU6050.h>
/*Потім треба створити змінну з ім'ям і типом датчика: MPU6050. Ця змінна дасть можливість використовувати функції бібліотеки. */
MPU6050 sensor ;
/*Після цього треба визначити змінні, які використовуються для вимірювання даних датчика MPU6050. */
int16_t ax, ay, az ; //дані акселерометра
int16_t gx, gy, gz ; //дані гіроскопа
/*Оператор підготовки setup () здійснює підключення зв'язку I2C та послідовного інтерфейса зі швидкістю 9600 бод/с. */
void setup (){
Wire.begin ( );
Serial.begin  (9600);
/*Наступним кроком є перевірка  роботи датчика MPU6050. Якщо він працює нормально на монітор виводиться повідомлення "Taking Values from the sensor" («Приймати значення від датчика»). Якщо він не підключен, то з'явиться  повідомлення “Connection failed" («Помилка з'єднання». */
Serial.println  ( "Initializing the sensor" );
sensor.initialize ( );
Serial.println (sensor.testConnection ( ) ? "Successfully Connected" : "Connection failed");
delay (1000);
Serial.println ( "Taking Values from the sensor" );
delay (1000);
}
/*Функція циклу loop () здійснює зчитування даних датчика MPU6050 і вивід на монітор. */
void loop (){
sensor.getMotion6 (&ax, &ay, &az, &gx, &gy, &gz); //функція зчитування даних
Serial.print(ax);   //вивід даних на монітор
Serial.print("\t");
Serial.print(ay);
Serial.print("\t");
```

```
Serial.print(az);
Serial.print("\t");
Serial.print(gx);
Serial.print("\t");
Serial.print(gy);
Serial.print("\t");
Serial.println(gz);
delay (200);
}
```

На рис. 5 наведені дані, що отримані за допомогою монітора та плотера інтегрованої середи розробки Arduino IDE.




а)




б)

Рис. 5. Дані, що отримані за допомогою монітора та плотера

Треба визначити, як пов’язані отримані дані з орієнтацією датчика для координати ах, (рис. 6). Якщо повернути акселерометр так, щоб його вісь опинилася в вертикальному положенні, то вантаж зміститься вниз, розтягнувши при цьому верхню пружину і стиснувши

нижню. У цей момент акселерометр зафіксує величину прискорення вільного падіння — 9.8 м/с².

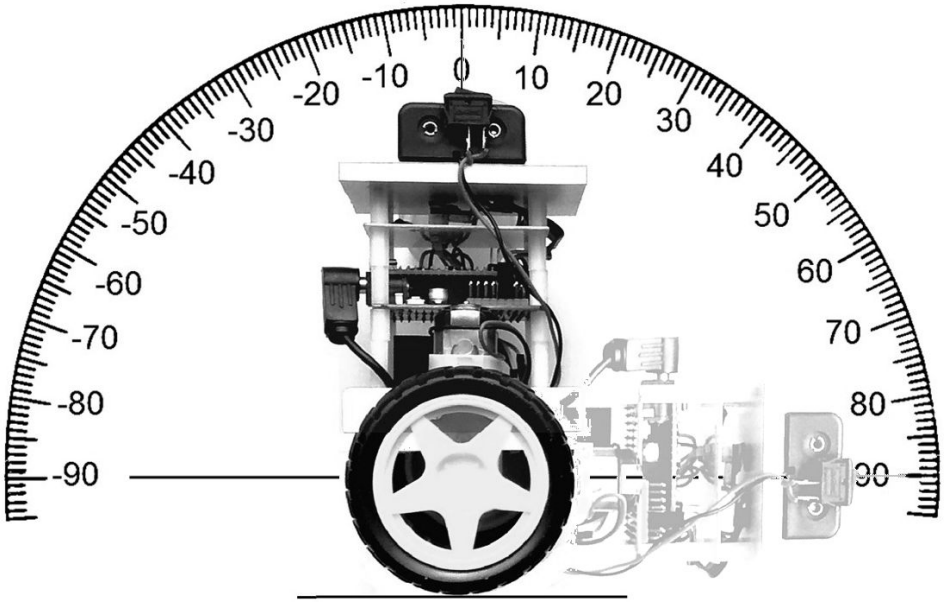



Рис. 6. Визначення коефіцієнти Кх

Знайти коефіцієнти Кх, для перерахування значень ах, маючи на увазі, що для кута повороту 90° це значення має дорівнювати 1.

Оскільки зворотні тригонометричні функції видають значення у радіанах, то програма,що визначає кути обертання у градусах має такий вигляд:

```
if( ax >= 0){
      angle_ax = 90 - RAD_TO_DEG*acos(ax);
   }
   else {
      angle_ax = RAD_TO_DEG*acos(-ax) - 90;
   }
```

Тут константа перерахунку радіанів у градусі є стандартною для мови С++ та має таке значення: RAD_TO_DEG = 57.2957795130823208767981548141051703 3f.

Виходячи з додатку 1 у разі використання гіроскопа, кут нахилу пристрою обчислюється за допомогою дискретного інтегрування швидкості його обертання.

$$a(t) = a(t-1) + gx*dt$$

де a (t) - шуканий кут нахилу тіла;
a (t-1) - кут тіла в попередній момент часу;

Об'єднання даних акселерометра та гороскопа дозволяє комплементарний фільтр, за допомогою якого отримується підсумкова величина кута нахилу є сумою інтегрованого значення гіроскопа і миттєвого значення акселерометра.

Програма, яка здійснює таке обчислювання має такий вигляд:

```
// обчислюємо кут нахилу по акселерометру
angle_ax = 90 - RAD_TO_DEG*acos(ay);
// обчислюємо кут нахилу по гироскопу
angle_gx = angle_gx + gx * T_OUT/1000.0;
// коригуємо значення кута за допомогою акселерометра
```

angle_gx = angle_gx*(1-FK) + angle_ax*FK;

Тут FK - коефіцієнт комплементарного фільтра,
angle_ax - кут нахилу по акселерометру,
angle_gx - кут нахилу з корекцією.

Програма **DataMPU6050_compl_filt_X**, що визначає кути обертання по осі X з використанням акселерометра, гіроскопа та корекцією за допомогою комплементарного фільтра наведена у додатку 7. На рис. 7 наведені результати, отримані на плотарі.

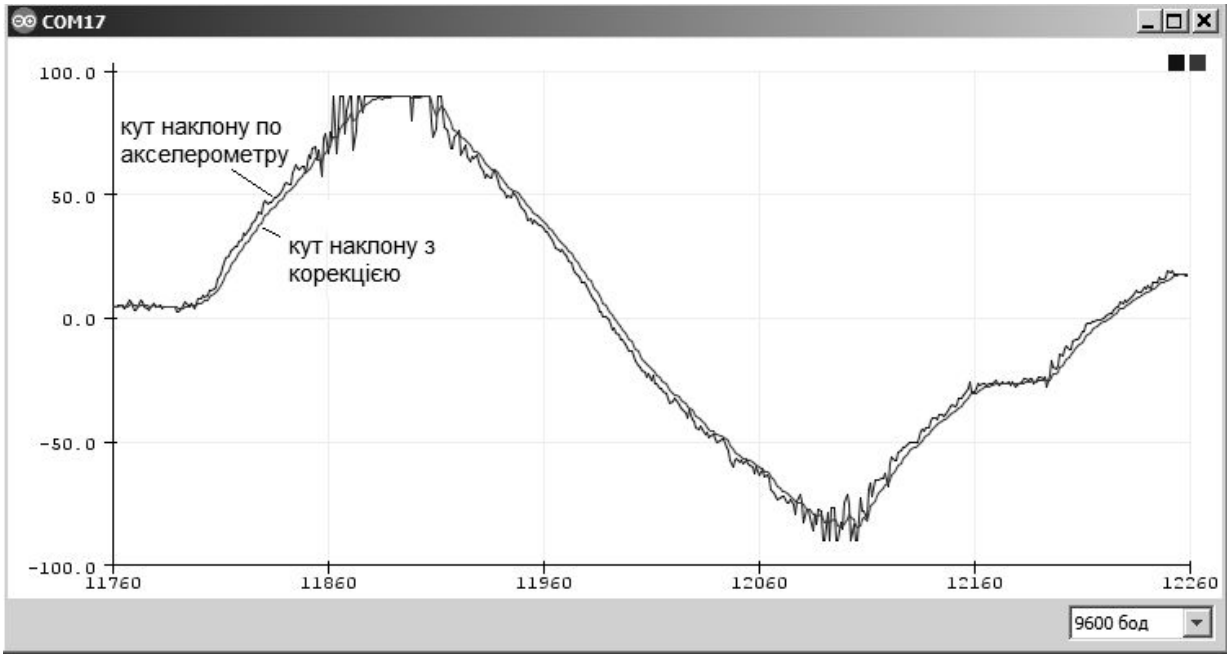



Рис. 7. Результати визначення значень angle_ax та angle_gx, отримані на плотарі

Визначаємо, як пов'язані отримані дані з кута нахила датчика з отриманими значеннями angle_ax (кут нахилу по акселерометру) та angle_gx ( кут нахилу з корекцією) (рис. 8).

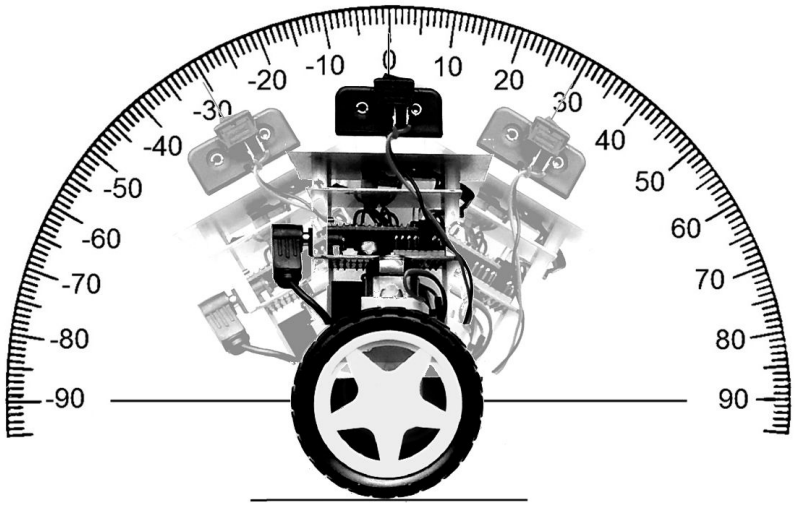



Рис. 8 Визначення даних з орієнтацією датчика

Отримані дані занести в таблицю.

| Кут повороту ° по осі x | -90 | -60 | -45 | -30 | 0 | 30 | 45 | 60 | 90 |
|---|---|---|---|---|---|---|---|---|---|
| angle_ax | | | | | | | | | |
| angle_gx | | | | | | | | | |

**3.1. Дослідження робота, який балансує**

Отримані результати дають можливість скласти програму для утримання балансуючого робота.

Для цього треба отриманий кут орієнтування по осі X порівняти з заданим значенням та здійснити керування приводами коліс, яке буде відхиляти балансуючий робот, як це показано на рис. 1.

Для керування приводами використовується

У додатку 8 наведений приклад програми, що здійснює балансування робота.

**Завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів орієнтації робота за допомогою акселерометра та гіроскопа.
2. Скласти програми для обчислення даних акселерометра та гіроскопа.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення даних акселерометра та гіроскопа на стенді.

**Завдання 2**

1. Розглянути теоретичні положення, а саме, алгоритмом, що забезпечує балансування двоколісного робота.
2. Ознайомитись з програмою, що забезпечує балансування двоколісного робота.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми, що забезпечує балансування двоколісного робота.
2. Визначити вплив параметрів ПІД-регулятора на стійкість двоколісного робота.

**Контрольні питання**

1. Визначити, які засоби дозволяють визначити орієнтацію двоколісного мобільного робота?
2. Описати, як визначити орієнтацію робота за допомогою гіроскопа-акселерометра MPU6050?
3. Визначити, які засоби потребуються для створення програми обчислення параметрів?
4. Описати, для чого використовуються комплементарний та інші фільтри?
5. Описати, для чого використовуються ПІД-регулятор?

**ПРОТОКОЛ**
**до лабораторного заняття № 4**
**Дослідження можливості використання комп'ютерних розрахунків для дослідження робота, який балансує**

**Студент _____ група _____**

**Мета заняття:** Дослідження можливості використання комп'ютерних розрахунків для балансування двоколісного мобільного робота за допомогою гіроскопа-акселерометра MPU6050

**Опис лабораторної установки**

Для визначення вказаних параметрів використовується лабораторний стенд. як це показано на рис. 1.

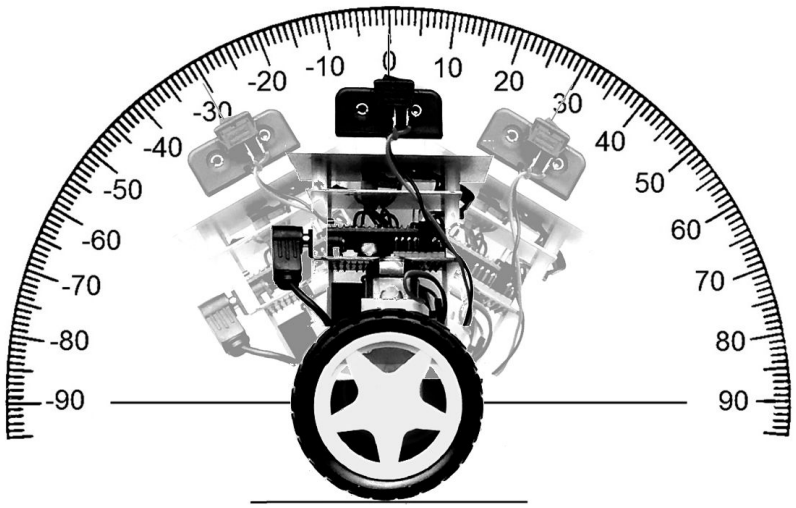



Рис. 1. Лабораторний стенд

**Завдання 1**

1. Розглянути теоретичні положення, а саме, з алгоритмом обчислювання параметрів орієнтації робота за допомогою акселерометра та гіроскопа.
2. Скласти програми для обчислення даних акселерометра та гіроскопа.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми для обчислення даних акселерометра та гіроскопа на стенді.
   Визначаємо, як пов'язані отримані дані з кута нахила датчика з отриманими значеннями angle_ax (кут нахилу по акселерометру) та angle_gx ( кут нахилу з корекцією). Отримані дані занести в таблицю.

| Кут повороту ° по осі x | -90 | -60 | -45 | -30 | 0 | 30 | 45 | 60 | 90 |
|---|---|---|---|---|---|---|---|---|---|
| angle_ax | | | | | | | | | |
| angle_gx | | | | | | | | | |

**Завдання 2**

1. Розглянути теоретичні положення, а саме, алгоритмом, що забезпечує балансування двоколісного робота.

2. Ознайомитись з програмою, що забезпечує балансування двоколісного робота.

**Опис ходу роботи (експерименту)**

1. Продемонструвати результати виконання програми, що забезпечує балансування двоколісного робота.
2. Визначити вплив параметрів ПІД-регулятора на стійкість двоколісного робота.

**Контрольні питання**

1. Визначити, які засоби дозволяють визначити орієнтацію двоколісного мобільного робота?

_____
_____
_____
_____
_____

2. Описати, як визначити орієнтацію робота за допомогою гіроскопа-акселерометра MPU6050?

_____
_____
_____
_____
_____

3. Визначити, які засоби потребуються для створення програми обчислення параметрів?

_____
_____
_____
_____
_____

4. Описати, для чого використовуються комплементарний та інші фільтри?

_____
_____
_____
_____
_____

5. Описати, для чого використовуються ПІД-регулятор?

_____
_____
_____
_____
_____

**Висновки**

_____
_____
_____
_____

**РЕКОМЕНДОВАНА ЛІТЕРАТУРА**


1. Навчальний посібник з дисципліни Маніпулятори та промислові роботи. Для студентів бакалаврів, спеціальності: 131 - Прикладна механіка, 133 – Галузеве машинобудування, / Укладачі: Михайлов Є. П., Лінгур В.М. – Одеса: ОНПУ, 2019. - 233 с. Рег ном. МВ09960 23.11.2018, №6223-РС-2018 URL: http://memos.library.opu.ua:8080/memos/jsp/materials.iface?mId=37024
2. Arduino довідник мови. Мова програмування Ардуіно - гнучкість і простота. Опис Arduino IDE URL: https://gamessoon.ru/uk/arduino-reference-language-the-programming-language-is-arduino-flexibility-and-simplicity/
3. Навчальний посібник з дисципліни «Машини і обладнання складів і логістика» для студентів магістрів, спеціальності: 131 - Прикладна механіка, спеціалізація: – Інженерія логістичних систем, / Укл.: Михайлов Є. П., Вудвуд О. М. – Одеса: ОНПУ, 2019. - 227с. URL: http://dspace.opu.ua/jspui/handle/123456789/9457
4. Цвіркун Л.І. Робототехніка та мехатроніка: навч. посіб. / Л.І. Цвіркун, Г. Грулер ; під заг. ред. Л.І. Цвіркуна ; М-во освіти і науки України, Нац. гірн. ун-т. –3-тє вид., переробл. і доповн. – Дніпро: НГУ, 2017. – 224 с. URL: http://ir.nmu.org.ua/handle/123456789/152141?show=full


ІНФОРМАЦІЙНІ РЕСУРСИ


5. Download the Arduino IDE URL: https://www.arduino.cc/en/Main/Software **(**Last accessed: 27.01.2020).
6. Arduino Self-Balancing Robot**:** URL: https://www.instructables.com/id/Arduino-Self-Balancing-Robot-1/ **(**Last accessed: 10.04.2020).
7. kurimawxx00/arduino-self-balancing-robot URL: https://github.com/kurimawxx00/arduino-self-balancing-robot **(**Last accessed: 10.04.2020).
8. Программирование Ардуино. Справочник языка Ардуино. Arduino UA. URL: https://doc.arduino.ua/ru/prog/#Structure **(**Last accessed: 10.04.2020).

**Додаток 1 Математичні функції та засоби обміну даними**

**Математичні функції**

**Складання, віднімання, множення і ділення**

Оператори +, -, * і / відповідно, повертають результат виконання арифметичних дій над двома операндами. Результат, що повертається, буде залежати від типу даних операндів, наприклад, 9/4 поверне 2, тому що операнди 9 і 4 мають тип int. Також слід стежити за тим, щоб результат не вийшов за діапазон допустимих значень за видом даних. Так, наприклад, складання 1 зі змінною типу int і значенням 32 767 поверне -32 768. Якщо операнди мають різні типи, то тип з більш "широким" діапазоном буде використаний для обчислень.

Якщо один з операндів має тип float або double, то арифметика "з плаваючою комою" буде використана для обчислень.

Синтаксіс:

result = value1 + value2;
result = value1 - value2;
result = value1 * value2;
result = value1 / value2;

Параметри:

value1: будь-яка змінна або константа;
value2: будь-яка змінна або константа.

**Квадратний корінь числа.**

sqrt(x)

Параметри:

x: число, будь-який тип даних.

Значення, що повертаються:

double, квадратний корінь числа.

**Квадрат числа.**

sq(x)

Параметри:

x: число, будь-який тип даних.

Значення, що повертаються:

квадрат числа.

**Зворотні тригонометричні функції**

asin(x), acos(x), atan(x).

Параметри:

x: число, тип даних float.

Значення, що повертаються:

зворотна тригонометрическая функція у радіанах, тип даних float.

**Приклад програми виконання арифметичних дій**

Створимо програму для послідовності арифметичних дій з отриманням результату у форматі float (з плаваючою комою) згідно з формулою:

$$z = x / ( x^2 + y^2)^{1/2}$$

```
sketch_arithmetic_20_02_25

float Var1; //вихідні дані X
float Var2; //вихідні дані  Y
float Var3; //результат Z

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

```
void loop() {
// Коли в буфері є дані, які були введені з монітору
while (Serial.available() > 0)
{
Var1 = Serial.parseFloat(); // Отримане число
Var2 = Serial.parseFloat(); // Отримане число
if (Serial.read() == '\n') // Кінець передачі
{
Var3 = Var1 /(sqrt(sq(Var1) + sq(Var2)));
Serial.print("X = \t");
Serial.println(Var1, 4); //вихідні дані X
Serial.print("Y = \t");
Serial.println(Var2, 4); //вихідні дані  Y
Serial.print("Z = \t");
Serial.println(Var3, 4); //результат Z
}
}
}
```

На рис. 1 наведені дані, що були отримані на моніторі за допомогою утиліти Монітор порту при обчисленні наведеної вище функції для значень x = 45, y = 56 (результат z =0,62639) . Числа, які вводяться у полі для введення даних, розділяються комою.

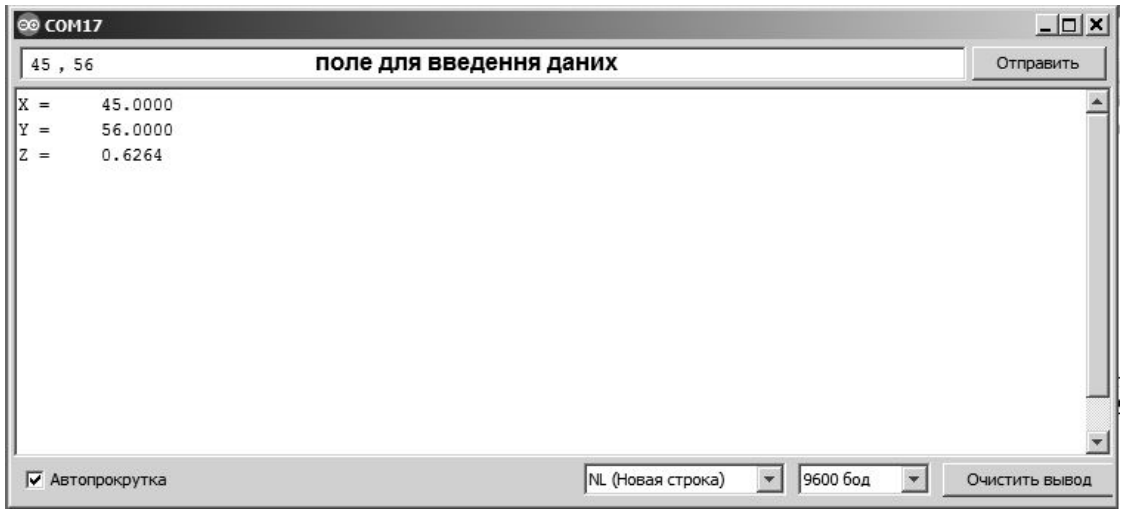



Рис. 1. Результат, отриманий на моніторі

**Засоби обміну даними**

Монітор порту Ардуіно та Плотер по послідовному з'єднанню - це утиліти, які вбудовані в середу програмування Arduino IDE і служить для зв'язку комп'ютера з контролером.

Монітор порту видає дані у символьному вигляді.

Плотер по послідовному з'єднанню видає дані у графичному вигляді.

Для відкриття утиліти Монітор порту необхідно натиснути на іконку в правому верхньому куті Arduino IDE, використовувати комбінацію клавіш Ctrl + Shift + M або вибрати в панелі меню: Инструменты > Монитор порта.

Для відкриття утиліти Плотер по послідовному з'єднанню необхідно вибрати в панелі меню: Инструменты > Плоттер по последовательному соединению (рис. 2).

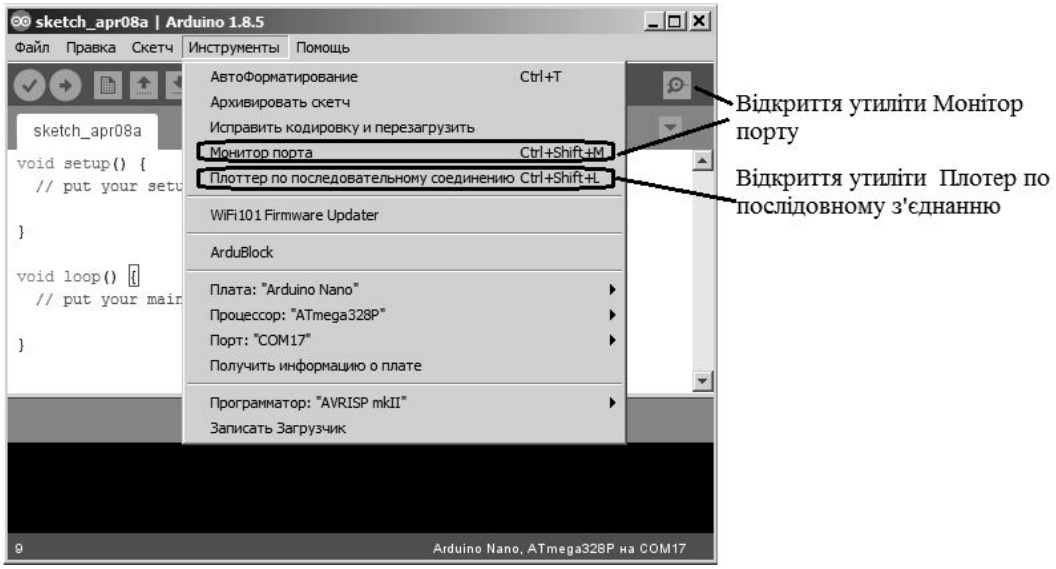



Рис. 2. Відкриття утиліт Монітор порту та Плотер по послідовному з'єднанню

**Монітор порту Ардуіно**

Для роботи з утилітою, використовують такі команди:

Serial.begin (); - команда запускає послідовний порт
Serial.end (); - зупиняє і очищає послідовний порт
Serial.print (); - відправляє дані в послідовний порт
Serial.print(78) - відправляє "78"
Serial.print(1.23456) - відправляє "1.23"
Serial.print(1.23456, 0) - відправляє "1"
Serial.print(1.23456, 2) - відправляє "1.23"
Serial.print(1.23456, 4) - відправляє "1.2346"
Serial.print('N') - відправляє "N"
Serial.print("Hello world.") - відправляє "Hello world."
Serial.print("\t"); - табуляція
Serial.println (); - відправляє дані з перенесенням рядка
Serial.read (); - приймає дані з послідовного порту
Serial.parseInt (); - читання цілих чисел з монітора
Serial.parseFloat(); - читання чисел з плаваючою точкою з монітора
Serial.available() - повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту

Введення даних з монітору здійснюється за допомогою функції:

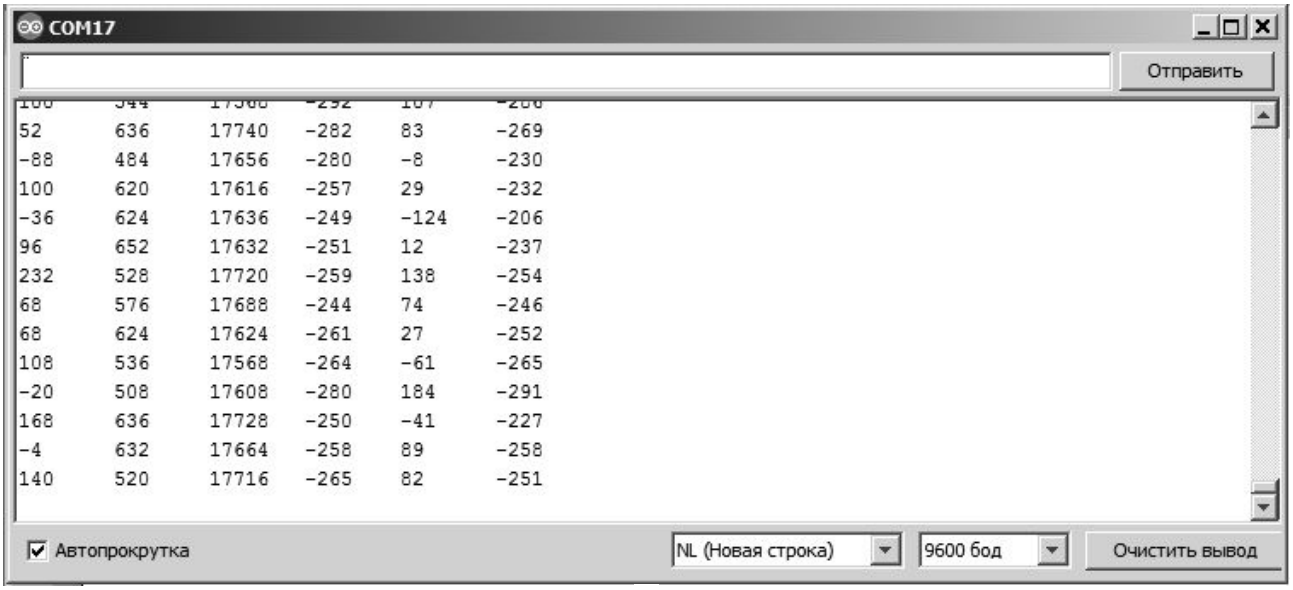
```
while (Serial.available() > 0)
{
Var1 = Serial.parseFloat(); // Перше отримане число
Var2 = Serial.parseFloat(); // Друге отримане число
if (Serial.read() == '\n') // Кінець передачі
{
Var3 = sqrt(sq(Var1) + sq(Var2)); //обчислення кореня квадратного з суми квадратів
…
}
```

**Плотер по послідовному з'єднанню**

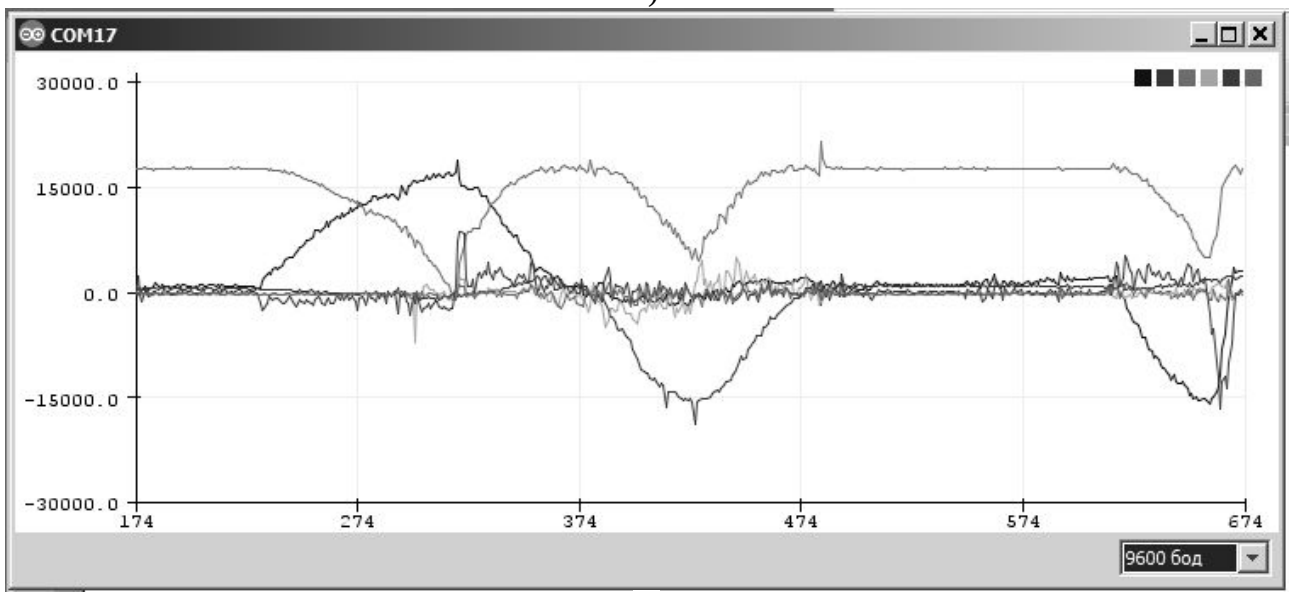
Плотер по послідовному з'єднанню здійснює виведення даних з контролера на монітор комп’ютера у графічному вигляді.

Для виведення даних на Плотер по послідовному з'єднанню використовуються команди Serial.print (); для кожної змінної, що поділяються командою табуляції Serial.print("\t"); . Для виведення останньої змінної використовуються команда Serial.println ();.

На рис. 3 наведений приклад виведення даних на Монитор порта та Плотер по послідовному з'єднанню.




а)




б)

Рис. 3. Приклад виведення даних на Монитор порта (а) та Плотер по послідовному з'єднанню (б)

**Додаток 2 Приклад програми обчислення параметрів для керування маніпулятором на основі зворотної задачі кінематики**

Приклад програми, що здійснює обчислення параметрів для керування маніпулятором на основі зворотної задачі кінематики за допомогою таких формул:

Phi1 = arccos( X/ L12 ) + arccos( ( $L1^2$ - $L2^2$ + $L12^2$ ) / 2· L12·L1 );

Phi2 = PI - arccos( ($L1^2$ + $L2^2$ - $L12^2$ ) / 2· L1·L2 )

**sketch_kinem_inv_example**

```
float VarX = 100; //вихідні дані X
float VarY = 0; //вихідні дані  Y
float  VarPhi1 =0;
float  VarPhi2 =0;
float  VarL1 =100;
float  VarL2 =100;
float  VarL =0;
float VarLL =0;
float  VarPhi1Deg = 0;
float  VarPhi2Deg = 0;
float  K1 = 0;
float  K2 = 0;
float  acosA = 0;
float  acosB = 0;
float  acosC = 0;
void setup() {
K1 = sq(VarL1) - sq(VarL2);
K2 = sq(VarL1) + sq(VarL2);
Serial.begin(9600);
}
void loop() {
// Коли в буфері є дані
while (Serial.available() > 0) {
VarX = Serial.parseFloat(); // Отримане число
VarY = Serial.parseFloat(); // Отримане число
if (Serial.read() == '\n') // Кінець передачі
{
VarLL = sq(VarX) + sq(VarY);
VarL = sqrt(VarLL);
acosA = acos(VarX / VarL);
acosB = acos((K1 + VarLL) / (2* VarL1* VarL));
acosC = acos((K2 - VarLL) / (2* VarL1* VarL2));
VarPhi1 = acosA +  acosB;
VarPhi2 = PI - acosC;
VarPhi1Deg = VarPhi1 * RAD_TO_DEG; //результат у градусах
VarPhi2Deg = VarPhi2 * RAD_TO_DEG; //результат у градусах
//виведення даних на монітор
Serial.print("X \t");
Serial.println(VarX, 2); //вихідні дані
Serial.print("Y \t");
Serial.println(VarY, 2); //вихідні дані
Serial.print("Phi1 \t");
Serial.println(VarPhi1, 4); //результат у радіанах
```

```
Serial.print("Phi1Deg \t");
Serial.println((int)VarPhi1Deg); //результат у градусах
Serial.print("Phi2 \t");
Serial.println(VarPhi2, 4); //результат у радіанах
Serial.print("Phi2Deg \t");
Serial.println((int)VarPhi2Deg); //результат у градусах
delay(1000);
}
}
```

**Додаток 3 Приклад програми визначення орієнтації за допомогою лазерного сканера**

Приклад програми, що визначає орієнтацію за допомогою лазерного сканера шляхом обчислення таких формул:

$X= (L1^2 + L12^2 – L2^2) / 2\ L12;$

$Y = (L1^2 - X^2)^{1/2}$ ;

ar = 270° - a2 - arccos(Y / L2) .

**lab_laser_scanner**

```
float VarL12; //результат положення L12
float VarL1; //результат положення L1
float VarL2; //результат положення L2
float a1Deg;  //результат кута a1 у градусах
float a2Deg;  //результат кута a2 у градусах
float VarX; //X
float VarY; //Y
float a22Rad; //результат кута a22 у радіанах
float a22Deg; //результат кута a22 у градусах
float arDeg; //результат кута ar у градусах
void setup() {
Serial.begin(9600);
}
void loop() {
while (Serial.available() > 0)  {        // Коли в буфері є дані
VarL12 = Serial.parseFloat();          // Отримане число L12
VarL1 = Serial.parseFloat();           // Отримане число L1
VarL2 = Serial.parseFloat();           // Отримане число L2
a1Deg = Serial.parseFloat();           // Отримане число a1
a2Deg = Serial.parseFloat();           // Отримане число a2
if (Serial.read() == '\n') {                // Кінець передачі
VarX = (sq(VarL1) + sq(VarL12)-sq(VarL2))/(2*VarL12);
VarY = sqrt(sq(VarL1) - sq(VarX));
a22Rad= acos (VarY / VarL2);  //результат a22 у радіанах
a22Deg = a22Rad * 180.0 / PI; //результат a22 у градусах
arDeg = 270 - a22Deg - a2Deg;
Serial.print("L12 \t");
Serial.println(VarL12, 2); //вихідні дані L12
Serial.print("L1 \t");
Serial.println(VarL1, 2); //вихідні дані L1
Serial.print("L2 \t");
Serial.println(VarL2, 2); //вихідні дані L2
Serial.print("X \t");
Serial.println(VarX, 2); //результат X
Serial.print("Y \t");
Serial.println(VarY, 2); //результат Y
Serial.print("ar \t");
Serial.println(arDeg, 2); //результат ar
}
}
}
```

**Додаток 4 Принцип дії акселерометра-гіроскопа MPU6050**

MPU-60X0 - це 6-осевий пристрій, який поєднує тривісний гіроскоп, тривісний акселерометр та цифровий процесор руху (DMP) у одному корпусі.

MPU-60X0 оснащений трьома 16-бітовими аналого-цифровими перетворювачами (АЦП) для оцифрування виходів гіроскопа та три 16-бітні АЦП для оцифрування виходів акселерометра. Для точного відстеження як швидких, так і повільних рухів пристрій оснащений програмованим гіроскопом повномасштабного діапазону ± 250, ± 500, ± 1000 та ± 2000 ° / с (dps) та програмованим акселерометром у повному масштабі діапазон ± 2g, ± 4g, ± 8g та ± 16g.

MPU-60X0 працює від напруги живлення VDD 3,5 В.

Зв'язок з усіма пристрями здійснюється за допомогою шини I2C.

**Принцип роботи акселерометра.**

Якщо повернути акселерометр так, щоб його вісь опинилася в вертикальному положенні, то вантаж зміститься вниз, розтягнувши при цьому верхню пружину і стиснувши нижню. У цей момент акселерометр зафіксує величину прискорення вільного падіння — 9.8 м/с² (рис. 1).

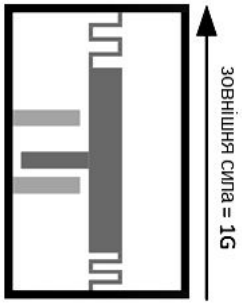



Рис. 1. Акселерометр фіксує величину прискорення вільного падіння

Цей факт використовуэться для обчислення кута нахилу акселерометра відносно горизонту. На схемі зображено тіло, де закріплений триосний акселерометр. Для вихідного положення означимо три осі як: Xт, Yт і Zт. Повернемо тіло на кут a навколо осі Xт щодо системи координат X, Y і Z. Передбачається, що вісь Z спрямована уздовж вектора сили гравітації (вгору), а осі X і Y уздовж горизонту (рис. 2).

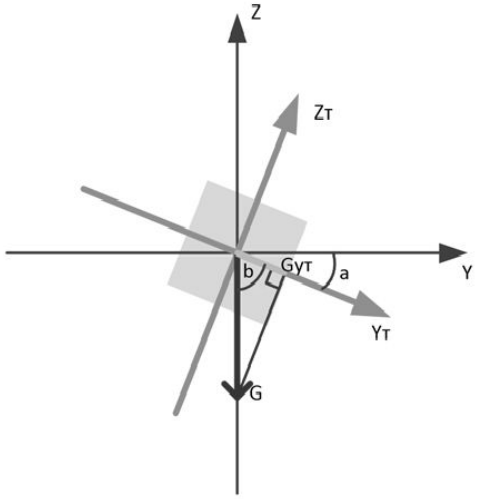



Рис. 2. Триосний акселерометр

a – кут нахилу тіла, b - кут нахилу тіла від осі Z, G вектор прискорення вільного паде, Gyt - проекція G на вісь тіла Yт

У такому положенні акселерометр, що знаходиться всередині тіла зафіксує проекції сили гравітації на всі три осі: Gxт, Gyт, Gzт. При цьому проекція Gxт на вісь Xт буде

дорівнює нулю, так як ця вісь розташована уздовж горизонту. Проекції Gут (зелений відрізок) і Gzт можна виразити за допомогою теореми про прямокутний трикутник

$$Gут = G * \cos(b) \quad (1)$$

$$Gzт = G * \sin(b) \quad (2)$$

Таким чином, знаючи G і одну з проекцій Gут або Gzт можна обчислити кут b відхилення акселерометра від вектора гравітації Z (від вертикальної осі):

$$\cos(b) = Gут/G \quad (3)$$

$$b = \arccos(Gут/G) \quad (4)$$

При такому обчисленні, важливо враховувати, що G і Gут повинні вимірюватися в однакових одиницях. Наприклад, якщо ми перетворимо показання акселерометра до одиниць гравітації (іншими словами G = 1 - земна гравітація), то вираз для кута b набуде вигляду:

$$b = \arccos(Gут/1) = \arccos(Gут) \quad (5)$$

Обчислимо кут a нахилу тіла відносно горизонту:

$$a = 90 - b = 90 - \arccos(Gут) \quad (6)$$

де Gут — це число, яке повертає нам акселерометр.

У разі використання гіроскопа, кут нахилу пристрою легко обчислюється за допомогою дискретного інтегрування швидкості його обертання.

$$a(t) = a(t-1) + gx*dt \quad (7)$$

де a (t) - шуканий кут нахилу тіла;

a (t-1) - кут тіла в попередній момент часу.

## Додаток 5 Алгоритм роботи фільтрів, які згладжують шуми

### Комплементарний фільтр

У гіроскопа є недолік, який називається дрейфом нуля. Суть цього недоліку зводиться до того, що при зупинці обертання гіроскопа, він все ще буде показувати значення відмінне від нуля.

Іншим недоліком такого рішення, є застосування процедури дискретного інтегрування, яка за своєю природою дає неточний результат.

Третя проблема виражається в поступовому накопиченні помилки обчислення кута через обмежену точність змінних мікроконтролера.

Таким чином є два прилади, кожен з яких дозволяє розрахувати кути нахилу машини відносно поверхні землі. Але в разі гіроскопа точність таких розрахунків знижується через дрейф нуля і помилку інтегрування. У разі ж акселерометра занадто є велика чутливість до зовнішніх впливів. Можна об'єднати показання цих двох пристроїв для усунення їх недоліків. Зробити таке об'єднання дозволяє комплементарний фільтр, який має таку формулу для інтегрування показань гіроскопа:

$$a(t) = (1-K) * (a(t-1) + gx*dt) + K * acc \quad (8)$$

де a (t) - шуканий кут нахилу, що враховує показання акселерометра;
a (t-1) - кут тіла в попередній момент часу;
gx - швидкість обертання тіла навколо осі X;
dt - час, який минув з моменту попереднього обчислення кута a;
acc - значення кута нахилу, отримане за допомогою акселерометра;
K - коефіцієнт комплементарного фільтра.

Як видно з формули, підсумкова величина кута нахилу є сумою інтегрованого значення гіроскопа і миттєвого значення акселерометра. По суті, головне завдання комплементарного фільтра тут в тому, щоб за допомогою показань акселерометра нівелювати дрейф нуля гіроскопа і помилки дискретного інтегрування.

### Фільтр Калмана

Фільтр Калмана - це рекурсивний фільтр, який оцінює поточний стан динамічної системи, використовуючи дані, що містять перешкоди. Фільтр Калмана успішно згладжує скачки, перетворюючи показання про стан системи до передбачуваних величин. Він враховує попередні свідчення і генерує згладження поточного значення.

Алгоритм розбитий на два етапи.

Перший етап - прогнозування, фільтр здійснює екстраполяцію значення змінних стану і їх невизначеності.

Під час другого етапу результат уточнюється на підставі отриманих вимірювань.

У разі використання фільтра Калмана у програмі треба визначити відповідну бібліотеку Kalman.h.

**Додаток 6 Алгоритм роботи PID-регулятора**

Пропорціонально-інтегрально-диференціальний (ПІД) регулятор формує керуючий сигнал, який є сумою трьох складових: пропорційної, інтегральної і диференціальної.

Рівняння ПІД-закону має такий вигляд:

$$Output = Kp \cdot e(t) + Ki \int e(t) \cdot dt + Kd \cdot \frac{d}{dt} e(t)$$

де *e(t) = setpoint – input.*

У цифровій формі це виглядає так:

*Output = Kp*(error) + Ki*(errorSum)*dt +Kd*(currentError – previousError)/dt = Kp * (error) + Ki *(errorSum)*sampleTime + Kd * (currentError – - previousError) / sampleTime*

де sample Time - час вибірки.

Пропорційна складова протидіє відхиленню регульованої величини від заданого значення. Для нашої системи регульована величина - це кут нахилу робота.

Інтегральна складова вносить в керуючий сигнал відповідь на накопичену помилку. По суті, це сума всіх помилок, помножених на період вибірки.

Диференціальна складова пропорційна швидкості зміни відхилення регульованої величини. Це різниця між поточною помилкою і попередньої помилкою, поділена на період вибірки. Ця складова призначена для протидії відхилень від цільового значення, яке прогнозується в майбутньому.

Помноживши кожну з цих складових на відповідні константи (Kp, Ki і Kd) і підсумовуючи результат, ми формуємо вихідний сигнал, який потім відправляється в якості команди для управління двигуном.

Для балансування використовується ПІД-регулятор, який реалізується за допомогою бібліотеки PID.

**Функції бібліотеки PID:**

**Функція PID()** створює ПІД-регулятор, пов'язаний із зазначеним входом, виходом і заданим значенням. Алгоритм PID виконується в паралельній формі.

Ця функція має такий синтаксис

PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)

Параметри функції PID:

Input: Змінна, яку ми намагаємося контролювати (double);

Output: Змінна, яка буде скоригована pid. (double);

Setpoint: Змінна, яка буде налаштована за допомогою pid (double);

Kp, Ki, Kd: Параметри налаштування. вони впливають на те, як pid змінить вихід. (double>=0);

Direction: DIRECT або REVERSE. визначає, в якому напрямку буде переміщуватися вихід, якщо зіткнутися з заданою помилкою. DIRECT є найбільш поширеним.

**Додаток 7 Програма, що визначає кути обертання**

Далі наведена програма, що визначає кути обертання з використанням акселерометра, гігрометра та здійснює корекцією за допомогою комплементарного фільтра.

**DataMPU6050_compl_filt_X**

```
//підключення бібліотек
#include "I2Cdev.h"
#include "MPU6050.h"
//визначення даних
#define T_OUT 20 // обчислення кожні 20 мілісекунд
#define P_OUT 50 // вивід даних кожні 50 мілісекунд
#define FK 0.1 // коефіцієнт комплементарного фільтра
MPU6050 accelgyro;
float angle_ax, angle_gx, angle_cpl;
float angle_x;
int dt = 0;
long int t_next, p_next;
// функція, що не дає значенню вийти за межи
float clamp(float v, float minv, float maxv){
   if( v>maxv )
      return maxv;
   else if( v<minv )
      return minv;
   return v;
}
void setup() {
   Serial.begin(9600);
   // инициализация MPU6050
   accelgyro.initialize();
   Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050
connection failed");
}
void loop() {
   long int t = millis();
   // кожні T_TO мілісекунд здійснюємо розрахунок кута наклону
   if( t_next < t ){
      int16_t ax_raw, ay_raw, az_raw, gx_raw, gy_raw, gz_raw;
      float ax,gx;
      t_next = t + T_OUT;
      // отримаємо дані від датчиків в MPU6050
      accelgyro.getMotion6(&ax_raw, &ay_raw, &az_raw, &gx_raw, &gy_raw, &gz_raw);
      // перетворюємо дані гіроскопа в град/сек
      gx = gx_raw / 16.4;
            // перетворюємо дані акселерометра в G
      ax = ax_raw / 16134.0;
      ax = clamp(ax, -1.0, 1.0);
      // кут наклону по акселерометру
      angle_ax = 90 - RAD_TO_DEG*acos(ax);
      // кут наклону по гіроскопу
      angle_gx = angle_gx + gx * T_OUT/1000.0;
      // корекція значення кута за допомогою акселерометра та комплементарного фільтра
```

```
      angle_gx = angle_gx*(1-FK) + angle_ax*FK;
    }
    t = millis();
    // кожні P_OUT мілісекунд видаємо результат в COM порт
    if( p_next < t ){
      p_next = t + P_OUT;
      Serial.print(angle_ax); // кут наклону по акселерометру
      Serial.print("\t");
      Serial.println(angle_gx);//кут наклону з корекцією
    }
}
```

**Додаток 8 Приклади програм балансуючого робота**

У наведених варіантах програми використовуються наступні бібліотеки. Бібліотека PID, що реалізує ПІД-регулятор, бібліотека LMotorController, яка використовується для управління двома двигунами з модулем L298H, бібліотека I2Cdev і бібліотека MPU6050_6_Axis_MotionApps20, які призначені для читання даних з MPU6050.

Варіант 1 [6].

**balancing_v3**

```
#include <Wire.h>
#include "Kalman.h"
#include "PID_v1.h"
Kalman kalmanY;
uint8_t IMUAddress = 0x68;
/* IMU Data */
int16_t accX;
int16_t accY;
int16_t accZ;
int16_t gyroY;
double accYangle;
double gyroYangle = 180;
double compAngleY = 180;
double kalAngleY;
double in, out, setpoint;
//PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)
PID pid(&in, &out, &setpoint, 30, 410 , 0.8, DIRECT);
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
uint32_t timer;
void setup() {
Serial.begin(115200);
Wire.begin();
i2cWrite(0x6B,0x00); // Disable sleep mode
                                // Вимкнути режим сну
if(i2cRead(0x75,1)[0] != 0x68) { // Read "WHO_AM_I" register
                                // Читання регістру «WHO_AM_I» - «ХТО_Я»
Serial.print(F("MPU-6050 with address 0x"));
Serial.print(IMUAddress,HEX);
Serial.println(F(" is not connected"));
while(1);
}
kalmanY.setAngle(180);
pid.SetMode(AUTOMATIC);
pid.SetOutputLimits(-255,255);
pid.SetSampleTime(10);
setpoint = 184.8;
timer = micros();
```

```
}

double ang;
int pwm;
void loop() {
 /* Update all the values */
uint8_t* data = i2cRead(0x3B,14);
accX = ((data[0] << 8) | data[1]);
accY = ((data[2] << 8) | data[3]);
accZ = ((data[4] << 8) | data[5]);
gyroY = ((data[10] << 8) | data[11]);
/* Calculate the angls based on the different sensors and algorithm
Обчисліть кути на основі різних датчиків та алгоритму */
accYangle = (atan2(accX,accZ)+PI)*RAD_TO_DEG;
double gyroYrate = -((double)gyroY/131.0);
gyroYangle += gyroYrate*((double)(micros()-timer)/1000000);
//Фільтр Калмана
kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, (double)(micros()-timer)/1000000);
timer = micros();
//ПІД регулятор
in = kalAngleY;
pid.Compute();
//Керування двигунами
if (out < 0) {   //вперед
pwm = -1 * out;
digitalWrite(ENA, HIGH);   //ENA
digitalWrite(ENB, HIGH);   //ENB
analogWrite(IN1, 0); //IN1
analogWrite(IN2, pwm);      //IN2
analogWrite(IN3, 0); //IN3
analogWrite(IN4, pwm);      //IN4
} else {          //назад
pwm = out;
digitalWrite(ENA, HIGH);   //ENA
digitalWrite(ENB, HIGH);   //ENB
analogWrite(IN2, 0); //IN2
analogWrite(IN1, pwm);      //IN1
analogWrite(IN4, 0); //IN4
analogWrite(IN3, pwm);      //IN3
}
delay(1); // The accelerometer's maximum samples rate is 1kHz
            // Максимальна швидкість вибірки акселерометра - 1 кГц
}
//Функція запису по шині  I2C
void i2cWrite(uint8_t registerAddress, uint8_t data){
Wire.beginTransmission(IMUAddress);
Wire.write(registerAddress);
Wire.write(data);
Wire.endTransmission(); // Send stop
}
//Функція читання по шині  I2C вказаної кількості байтів
uint8_t* i2cRead(uint8_t registerAddress, uint8_t nbytes) {
uint8_t data[nbytes];
```

```
Wire.beginTransmission(IMUAddress);
Wire.write(registerAddress);
Wire.endTransmission(false);          // Don't release the bus
                                      // Не відпускайте шину
Wire.requestFrom(IMUAddress, nbytes);
// Send a repeated start and then release the bus after reading
// Надіслати повторний старт, а потім звільнити шину після читання
for(uint8_t i = 0; i < nbytes; i++)
data[i] = Wire.read();
return data;
}
```

Варіант 2 [7].

**balancing_v4_AmBOT_**

```
#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
   #include "Wire.h"
#endif
#define MIN_ABS_SPEED 20
MPU6050 mpu;
// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorFloat gravity;    // [x, y, z]            gravity vector
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
//PID
//double originalSetpoint = 175.8;
double originalSetpoint = 181.0;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
int moveState=0; //0 = balance; 1 = back; 2 = forth
double Kp = 50;
double Kd = 1.4;
double Ki = 60;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
double motorSpeedFactorLeft = 0.6;
double motorSpeedFactorRight = 0.5;
//MOTOR CONTROLLER
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
```

```
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft,
motorSpeedFactorRight);
//timers
long time1Hz = 0;
long time5Hz = 0;
volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady()
{
   mpuInterrupt = true;
}
void setup()
{
   // join I2C bus (I2Cdev library doesn't do this automatically)
   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
      TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
   #endif
   // initialize serial communication
   // (115200 chosen because it is required for Teapot Demo output, but it's
   // really up to you depending on your project)
   Serial.begin(115200);
   while (!Serial); // wait for Leonardo enumeration, others continue immediately
   // initialize device
   Serial.println(F("Initializing I2C devices..."));
   mpu.initialize();
   // verify connection
   Serial.println(F("Testing device connections..."));
   Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050
connection failed"));
   // load and configure the DMP
   Serial.println(F("Initializing DMP..."));
   devStatus = mpu.dmpInitialize();
   // supply your own gyro offsets here, scaled for min sensitivity
   mpu.setXGyroOffset(220);
   mpu.setYGyroOffset(76);
   mpu.setZGyroOffset(-85);
   mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
   // make sure it worked (returns 0 if so)
   if (devStatus == 0)
   {
      // turn on the DMP, now that it's ready
      Serial.println(F("Enabling DMP..."));
      mpu.setDMPEnabled(true);
      // enable Arduino interrupt detection
      Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
      attachInterrupt(0, dmpDataReady, RISING);
      mpuIntStatus = mpu.getIntStatus();
      // set our DMP Ready flag so the main loop() function knows it's okay to use it
```

```
        Serial.println(F("DMP ready! Waiting for first interrupt..."));
        dmpReady = true;
        // get expected DMP packet size for later comparison
        packetSize = mpu.dmpGetFIFOPacketSize();
        //setup PID
        pid.SetMode(AUTOMATIC);
        pid.SetSampleTime(10);
        pid.SetOutputLimits(-255, 255);
    }
    else
    {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code will be 1)
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }
}
void loop()
{
    // if programming failed, don't try to do anything
    if (!dmpReady) return;
    // wait for MPU interrupt or extra packet(s) available
    while (!mpuInterrupt && fifoCount < packetSize)
    {
        //no mpu data - performing PID calculations and output to motors

        pid.Compute();
        motorController.move(output, MIN_ABS_SPEED);
    }
    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();
    // get current FIFO count
    fifoCount = mpu.getFIFOCount();
    // check for overflow (this should never happen unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024)
    {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
    // otherwise, check for DMP data ready interrupt (this should happen frequently)
    }
    else if (mpuIntStatus & 0x02)
    {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
```

```
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        #if LOG_INPUT
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI);
        #endif
        input = ypr[1] * 180/M_PI + 180;
    }
}
```