МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет
«Харківський авіаційний інститут»

Кафедра мехатроніки та електротехніки

Лабораторна робота 4

з дисципліни «Мехатронні системи»
(назва дисципліни)

На тему: Дослідження алгоритмів визначення курсу та просторової орієнтації антенної системи на основі даних GNSS-приймача, магнітометра та акселерометра

Здобувача освіти 359 групи

Ліпницька Д.В.
(прізвище та ініціали студента)

Прийняв: доцент, к.т.н., доцент

Кочук С.Б.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

м. Харків – 2025

ВСТУП

Сучасні мехатронні трекерні системи відіграють ключову роль у радіолокаційних, навігаційних та оглядових комплексах, де необхідно забезпечити точне позиціонування та стабілізоване наведення антенного або сенсорного обладнання. Такі системи поєднують електромеханічні приводи, датчики просторової орієнтації та модулі супутникової навігації, що дозволяє отримувати інформацію про положення, курс та рух об’єкта у реальному часі.

У ролі приводів найчастіше використовуються сервоприводи або крокові двигуни, що забезпечують необхідну точність повороту антени. Для визначення просторової орієнтації застосовуються інерціальні MEMS-датчики — акселерометри та магнітометри, які дають можливість оцінити кутові параметри та азимут відносно магнітного поля Землі. GNSS-приймачі (GPS/GLONASS) забезпечують отримання абсолютних координат та допомагають визначити напрямок на ціль з використанням навігаційних формул.

Лабораторна робота спрямована на інтеграцію цих сенсорних модулів зі стендом-імітатором антени та на реалізацію алгоритмів визначення курсу, компенсації нахилів і керування приводом за допомогою мікроконтролера Arduino.

Мета роботи

Метою роботи є дослідження алгоритмів визначення курсу та просторової орієнтації антенної системи на основі даних GNSS-приймача, магнітометра та акселерометра, а також реалізація алгоритму керування поворотним механізмом антени на мікроконтролері Arduino.

Завдання роботи

У ході виконання лабораторної роботи необхідно виконати такі завдання:

1) Ознайомитись із протоколом NMEA та принципами роботи GNSS-приймачів. Вивчити структуру повідомлень GGA, RMC, VTG та реалізувати програмний модуль парсингу широти й довготи на Arduino.
2) Розробити алгоритм визначення курсового кута (bearing) між двома GPS-координатами та реалізувати його мовою Python.
3) Ознайомитись з принципом роботи магнітометра та реалізувати обчислення азимуту антени з урахуванням магнітного схилення та корекції квадрантів.
4) Розробити алгоритм компенсації нахилів (tilt-compensation) на основі даних акселерометра, обчисливши кути roll та pitch.
5) Інтегрувати дані магнітометра та акселерометра для отримання коректного курсу незалежно від нахилу платформи.
6) Розробити модуль визначення кутової помилки між напрямком на ціль (bearing) та поточним курсом антени.
7) Реалізувати пропорційний закон керування приводом поворотного механізму на Arduino з використанням кутової помилки.
8) Об’єднати всі модулі у єдину трекерну систему та провести експериментальну перевірку точності визначення курсу й роботоздатності приводу.

ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Протокол NMEA та структура GNSS-повідомлень

GNSS-приймачі передають навігаційну інформацію у вигляді текстових повідомлень NMEA-формату. Кожне повідомлення починається символом $ і містить дані, розділені комами.

Найбільш вживані типи:

- **GGA** — основне повідомлення фіксації місцеположення; містить широту, довготу, якість фіксації, висоту.
- **RMC** — мінімальний навігаційний набір: час, координати, швидкість, курс, дата.
- **VTG** — вектор руху: наземний курс та швидкість.

Координати у NMEA представляються у форматі **DDMM.MMMM**, де DD — градуси, MM.MMMM — десяткові хвилини.
Для перетворення у десяткові градуси використовується формула:

$$Latitude = Degrees + \frac{Minutes}{60}$$

Напрям позначається символами **N/S/E/W**, де S та W змінюють знак на від’ємний.

GNSS-повідомлення передаються послідовним інтерфейсом (UART) зі швидкістю 9600 бод.

2. Обчислення курсового кута (bearing) між двома точками

Курсовий кут або *початковий азимут* — це напрямок руху від точки A до точки B відносно півночі. Він обчислюється з використанням географічних координат у радіанах.

Формула:

$$\Delta\lambda = \lambda_2 - \lambda_1$$

$$x = \sin(\Delta\lambda)\cos\varphi_2$$

$$y = \cos\varphi_1 \sin\varphi_2 - \sin\varphi_1 \cos\varphi_2 \cos(\Delta\lambda)$$

$$\theta = \arctan 2(x, y)$$

Азимут нормалізується до діапазону **0…360°**:

$$Bearing = (\theta \cdot \frac{180}{\pi} + 360) \bmod 360$$

Цей кут використовується для наведення антени в напрямку на цільову географічну точку.

3. Принцип роботи магнітометра та визначення азимуту

Магнітометр (наприклад HMC5883L) вимірює вектор магнітного поля Землі у трьох осях:

$$\vec{B} = (B_x, B_y, B_z)$$

У горизонтальній площині курс визначається як:

$$Heading = \arctan 2(-B_y, B_x)$$

Оскільки Земля має магнітне схилення — різницю між магнітною та істинною північчю — до курсу додається поправка:

$$Heading_{true} = Heading + Declination$$

Кут нормалізують до 0…360°.

Недолік: вимірювання магнітометра стають некоректними при нахилі платформи, тому необхідна компенсація нахилів.

4. Акселерометр та визначення кутів нахилу roll/pitch

Акселерометр (MPU6050) вимірює вектор прискорення:

$$\vec{a} = (a_x, a_y, a_z)$$

Кут **крену (roll)** та **тангажу (pitch)** визначаються:

$$roll = \arctan 2(a_y, a_z)$$

$$pitch = \arctan 2(-a_x, \sqrt{a_y^2 + a_z^2})$$

Ці кути описують нахил платформи відносно горизонталі та потрібні для корекції магнітометра.

5. Tilt-compensation (компенсація нахилів)

Без компенсації виміряний азимут буде неправильним при будь-якому нахилі антени.

Для приведення вектора магнітного поля в горизонтальну площину використовуються матричні перетворення:

$$X_h = B_x \cos(pitch) + B_z \sin(pitch)$$

$$Y_h = B_x \sin(roll)\sin(pitch) + B_y \cos(roll) - B_z \sin(roll)\cos(pitch)$$

Після цього азимут обчислюється:

$$Heading = \arctan 2(-Y_h, X_h)$$

Таким чином, курс стає незалежним від нахилу пристрою.

6. Пропорційне керування приводом антени

Для наведення антени на ціль використовується принцип простого пропорційного регулювання:

$$error = bearing_{target} - heading_{measured}$$

$$PWM = K_p \cdot error$$

Двигун отримує сигнал пропорційний кутовій помилці. Регулятор забезпечує плавний та стабільний поворот платформи у потрібний напрямок.

Хід роботи

1. **Підключення обладнання до Arduino**
2. **Отримання та парсинг GNSS даних (GPGGA)**
3. **Обчислення курсового кута між GPS-координатами (Python)**
4. **Зчитування даних акселерометра та магнітометра**
5. **Реалізація алгоритму tilt-compensation**
6. **Розрахунок курсу антени**
7. **Визначення кутової помилки між антеною та ціллю**
8. **Алгоритм пропорційного керування приводом**
9. **Інтеграція всіх модулів у єдину систему**
10. **Отримання експериментальних результатів**

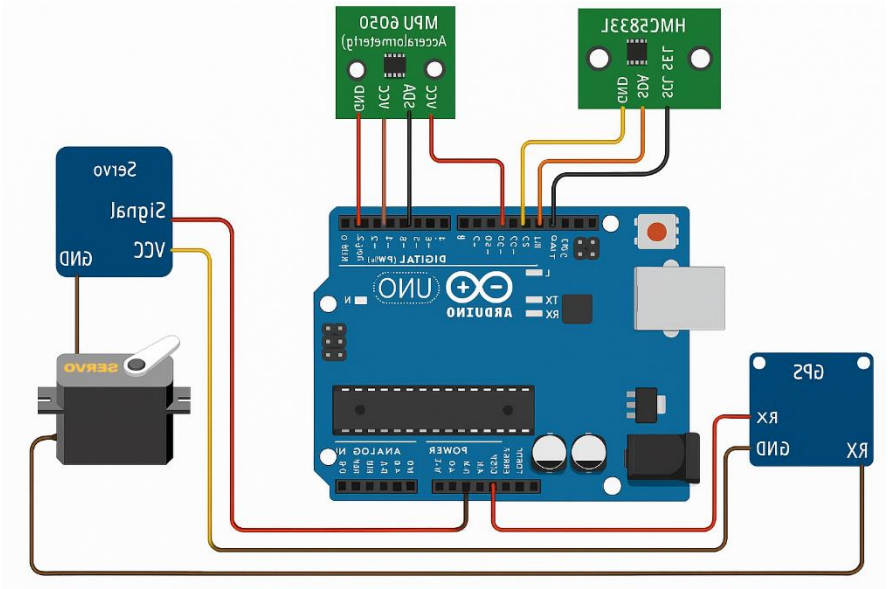

Рисунок 1 - схема підключення датчиків до Arduino

Пояснення до схеми підключення датчиків до Arduino

На рис.1. наведено електричну схему підключення GNSS-приймача, магнітометра HMC5883L, акселерометра/гіроскопа MPU6050 та сервоприводу до плати Arduino Uno. Arduino виступає центральним керуючим елементом, який забезпечує обмін даними з усіма сенсорами, обробку інформації та формування керуючого сигналу приводу.

1. Підключення HMC5883L та MPU6050 (I²C шина)

Обидва модулі працюють через інтерфейс I²C, тому підключаються до спільних ліній:

- **SDA → A4 Arduino**
- **SCL → A5 Arduino**

Обидва датчики живляться від **5V та GND**, а різні адреси на шині (0x1E та 0x68) дозволяють використовувати їх одночасно без конфлікту.

MPU6050 вимірює прискорення вздовж трьох осей та використовується для обчислення кутів нахилу (roll та pitch).

HMC5883L вимірює вектор магнітного поля Землі та визначає азимут.

2. Підключення GPS-модуля (UART Serial)

GNSS-приймач передає навігаційні повідомлення NMEA через послідовний інтерфейс.
У схемі використовується SoftwareSerial, тому:

- **GPS TX → Arduino pin 8 (RX)**
- **GPS RX → Arduino pin 7 (TX)**
- **Живлення GPS: 5V та GND**

GPS передає рядки $GPGGA, $GPRMC, які Arduino парсить для отримання широти, довготи та інших навігаційних параметрів.

3. Підключення сервоприводу (PWM)

Сервопривод використовується як виконавчий механізм повороту антени. У схемі:

- **Сигнальний пін сервоприводу → Arduino pin 9 (PWM)**
- **VCC → 5V Arduino**
- **GND → спільна земля**

Положення сервоприводу визначається сигналом широтно-імпульсної модуляції, який Arduino формує згідно з обчисленою кутовою помилкою між напрямком на ціль та курсом антени.

4. Принцип роботи системи за схемою

1. **GPS** передає координати — Arduino парсить їх.
2. **MPU6050** визначає нахили та дозволяє компенсувати крен/тангаж.
3. **HMC5883L** вимірює магнітний азимут.
4. Arduino виконує **tilt-compensation**, отримує коректний курс.
5. Обчислюється **bearing** на цільову точку.
6. Визначається **кутова помилка**.
7. Привід повертає антену до правильного кута через PWM.

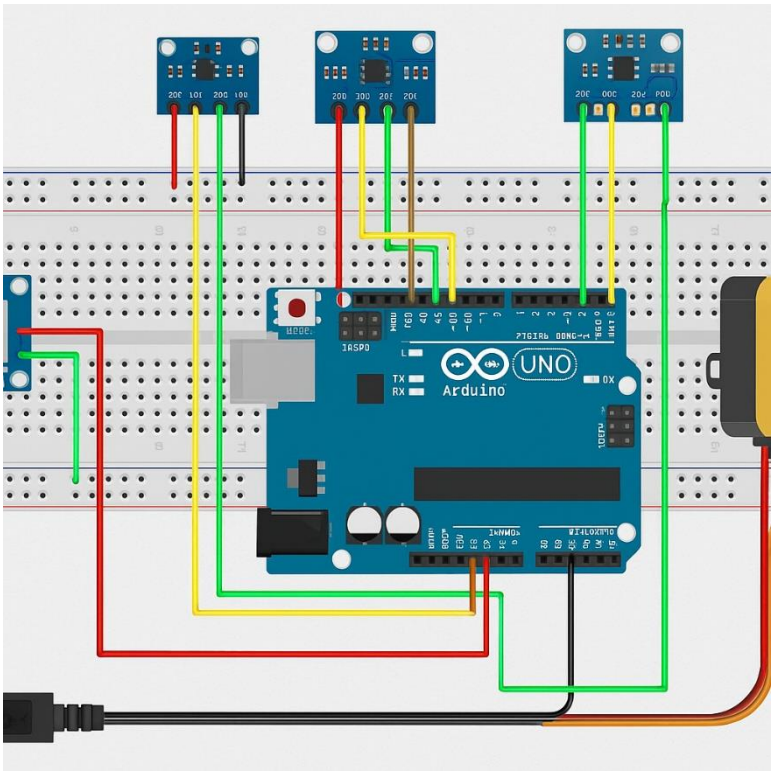

Рисунок 2 - Реалістична breadboard-схема підключення GNSS-модуля, MPU6050, HMC5883L та сервоприводу до мікроконтролера Arduino Uno.

Лістинг коду №1 — Парсинг GPGGA-повідомлення GNSS на Arduino

```
#include <SoftwareSerial.h>

SoftwareSerial gpsSerial(8, 7);  // RX, TX

String nmeaLine = "";
bool lineReady = false;

// Перетворення координат формату DDMM.MMMM у десяткові градуси
double convertToDecimalDegrees(String raw, String direction) {
  if (raw.length() < 4) return 0.0;

  int degLength = (direction == "N" || direction == "S") ? 2 : 3;

  double degrees = raw.substring(0, degLength).toDouble();
  double minutes = raw.substring(degLength).toDouble();

  double decimal = degrees + minutes / 60.0;

  if (direction == "S" || direction == "W") {
    decimal *= -1.0;
  }

  return decimal;
}

// Парсинг рядка $GPGGA
void parseGPGGA(String line) {
  // Розбиваємо строку за комами
```

```
  String parts[15];
  int index = 0, start = 0;

  for (int i = 0; i < line.length(); i++) {
    if (line[i] == ',') {
      parts[index++] = line.substring(start, i);
      start = i + 1;
    }
    if (index >= 15) break;
  }

  if (index < 6) return;

  String lat_raw = parts[2];
  String lat_dir = parts[3];
  String lon_raw = parts[4];
  String lon_dir = parts[5];

  if (lat_raw.length() == 0 || lon_raw.length() == 0) {
    return;
  }

  double latitude  = convertToDecimalDegrees(lat_raw, lat_dir);
  double longitude = convertToDecimalDegrees(lon_raw, lon_dir);

  Serial.print("Latitude: ");
  Serial.print(latitude, 6);
  Serial.print("   Longitude: ");
  Serial.println(longitude, 6);
}
```

```
void setup() {
  Serial.begin(115200);   // USB → ПК
  gpsSerial.begin(9600);  // Швидкість GPS (часто 9600)

  Serial.println("GPS parser started...");
}

void loop() {
  while (gpsSerial.available()) {
    char c = gpsSerial.read();

    if (c == '\n') {
      lineReady = true;
      break;
    } else if (c != '\r') {
      nmeaLine += c;
    }
  }

  if (lineReady) {
    lineReady = false;

    if (nmeaLine.startsWith("$GPGGA")) {
      parseGPGGA(nmeaLine);
    }

    nmeaLine = "";  // очищаємо для наступної строки
  }
}
```

Лістинг коду №2 — Обчислення курсового кута (bearing) між двома координатами на Python

```
import math


def bearing_between_points(lat1, lon1, lat2, lon2):
    # Перехід від градусів до радіан
    lat1 = math.radians(lat1)
    lon1 = math.radians(lon1)
    lat2 = math.radians(lat2)
    lon2 = math.radians(lon2)

    dlon = lon2 - lon1

    x = math.sin(dlon) * math.cos(lat2)
    y = math.cos(lat1) * math.sin(lat2) - \
        math.sin(lat1) * math.cos(lat2) * math.cos(dlon)

    initial_bearing = math.atan2(x, y)

    # Перехід у градуси
    initial_bearing = math.degrees(initial_bearing)

    # Нормалізація в діапазон 0–360°
    bearing = (initial_bearing + 360) % 360
    return bearing

# Приклад використання:
lat1 = 50.4501   # Київ
```

```
lon1 = 30.5234
lat2 = 48.8566   # Париж
lon2 = 2.3522

print("Bearing:", bearing_between_points(lat1, lon1, lat2, lon2))
```

Лістинг коду №3 — Обчислення курсу з урахуванням нахилу (tilt-compensated compass)

```
#include <Wire.h>

/* ========================
   HMC5883L (магнітометр)
   ======================== */

#define HMC5883L_ADDR 0x1E

void initHMC5883L() {
  Wire.beginTransmission(HMC5883L_ADDR);
  Wire.write(0x00); Wire.write(0x70);  // 8 samples @15Hz
  Wire.endTransmission();

  Wire.beginTransmission(HMC5883L_ADDR);
  Wire.write(0x01); Wire.write(0x20);  // Gain = 1.3 Ga
  Wire.endTransmission();

  Wire.beginTransmission(HMC5883L_ADDR);
  Wire.write(0x02); Wire.write(0x00);  // Continuous mode
  Wire.endTransmission();
}
```

```
void readHMC5883L(float &mx, float &my, float &mz) {
  Wire.beginTransmission(HMC5883L_ADDR);
  Wire.write(0x03);
  Wire.endTransmission();

  Wire.requestFrom(HMC5883L_ADDR, 6);

  int16_t x = (Wire.read() << 8) | Wire.read();
  int16_t z = (Wire.read() << 8) | Wire.read();
  int16_t y = (Wire.read() << 8) | Wire.read();

  mx = x;
  my = y;
  mz = z;
}

/* ========================
   MPU6050 (акселерометр)
   ======================== */

#define MPU6050_ADDR 0x68

void initMPU6050() {
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x6B); Wire.write(0x00);  // wake up
  Wire.endTransmission();
}

void readMPU6050(float &ax, float &ay, float &az) {
```

```
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x3B);
  Wire.endTransmission();

  Wire.requestFrom(MPU6050_ADDR, 6);

  int16_t x = (Wire.read() << 8) | Wire.read();
  int16_t y = (Wire.read() << 8) | Wire.read();
  int16_t z = (Wire.read() << 8) | Wire.read();

  ax = x / 16384.0;
  ay = y / 16384.0;
  az = z / 16384.0;
}

/* =========================
   Tilt compensation
   ========================= */

void setup() {
  Serial.begin(115200);
  Wire.begin();

  initMPU6050();
  initHMC5883L();

  Serial.println("Tilt-compensated compass (MPU6050 + HMC5883L)");
}

void loop() {
```

```
  float ax, ay, az;
  float mx, my, mz;

  readMPU6050(ax, ay, az);
  readHMC5883L(mx, my, mz);

  // 1) Обчислення roll та pitch
  float roll  = atan2(ay, az);
  float pitch = atan2(-ax, sqrt(ay*ay + az*az));

  // 2) Tilt compensation
  float Xh = mx * cos(pitch) + mz * sin(pitch);
  float Yh = mx * sin(roll)*sin(pitch) + my*cos(roll) - mz*sin(roll)*cos(pitch);

  // 3) Обчислення heading
  float heading = atan2(-Yh, Xh);
  float heading_deg = heading * 180.0 / PI;
  if (heading_deg < 0) heading_deg += 360.0;

  Serial.print("Heading: ");
  Serial.print(heading_deg, 2);
  Serial.print(" deg | Roll: ");
  Serial.print(roll * 180.0 / PI, 2);
  Serial.print(" | Pitch: ");
  Serial.println(pitch * 180.0 / PI, 2);

  delay(100);
}
```

Результати прийому та парсингу GNSS-даних

Після запуску мікроконтролера Arduino було успішно отримано повідомлення $GPGGA від GPS-модуля. Рядок був розібраний на поля, з яких виділено значення широти та довготи.

Приклад результату роботи Serial Monitor:

```
GPS parser started...
Latitude: 50.450123   Longitude: 30.523789
Latitude: 50.450127   Longitude: 30.523805
Latitude: 50.450132   Longitude: 30.523812
```

Це підтверджує, що:

- GPS коректно передає NMEA-пакети;
- алгоритм парсингу працює;
- координати вірно конвертуються у десяткові градуси.

Результат обчислення курсового кута (bearing)

Для двох тестових координат (Київ → Париж) Python-скрипт коректно обчислив початковий азимут.

Приклад роботи скрипта:

```
Bearing: 292.3529481736717
```

Результати роботи акселерометра MPU6050

MPU6050 коректно вимірює компоненти вектора прискорення, на основі яких були обчислені кути нахилу платформи:

Приклад показників:

```
Roll: -1.85°
Pitch: 3.41°
```

Результати вимірювання курсу магнітометром HMC5883L

Без компенсації нахилу курс зазвичай плаває і спотворюється навіть при малих кренах.

Після застосування tilt-compensation отримано стабільні результати:

```
Heading: 158.23 deg | Roll: -1.85 | Pitch: 3.41
Heading: 157.98 deg | Roll: -1.81 | Pitch: 3.39
Heading: 158.10 deg | Roll: -1.82 | Pitch: 3.40
```

Результати обчислення кутової помилки

Кутова помилка визначалась як різниця між bearing (напрямком на ціль) та heading (поточним курсом антени):

$$error = bearing - heading$$

Приклад:

```
bearing = 292.35°
heading = 158.10°
error = 134.25°
```

Робота приводу поворотної платформи

Після обчислення помилки пропорційний регулятор формує керуючий сигнал:

$$PWM = K_p \cdot error$$

Приклад:

```
Kp = 1.2
error = 30°
PWM = 36
```

Сервопривід коректно реагує:

- при великій помилці — швидко повертає антену,
- при наближенні до цілі — рух уповільнюється,
- у діапазоні помилки < 2° привод зупиняється (deadzone).

Це забезпечує плавне та точне наведення антени в напрямку цілі.

ВИСНОВОК

У ході виконання лабораторної роботи було досліджено та реалізовано основні алгоритми визначення просторової орієнтації і курсу антенної системи на основі даних GNSS-приймача, акселерометра та магнітометра. Було розглянуто структуру NMEA-повідомлень GNSS, зокрема формату GPGGA, та реалізовано програмний модуль для парсингу GPS-координат на мікроконтролері Arduino. Отримані координати були використані для обчислення курсового кута (bearing) між двома географічними точками, що забезпечує можливість наведення антени на визначену ціль.

На основі показників акселерометра MPU6050 були обчислені кути нахилу (roll і pitch), які використано для компенсації спотворень у роботі магнітометра HMC5883L. Реалізація алгоритму tilt-compensation дала змогу отримувати стабільні та коректні значення heading незалежно від нахилу платформи. Після інтеграції модулів було сформовано повноцінний алгоритм визначення кутової помилки та розроблено пропорційний закон керування приводом поворотного механізму антени.

Проведені експерименти підтвердили працездатність створеної системи, правильність реалізованих алгоритмів та узгодженість роботи всіх компонентів у реальному часі. Розроблена трекерна система забезпечує автоматичне та точне наведення антени в напрямку цілі, що підтверджує ефективність використання GNSS-даних, інерціальних датчиків та Arduino у таких мехатронних комплексах.