МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ ТРЕКЕРА СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ ДЛЯ АВТОНОМНИХ БЛОКПОСТІВ ПОЛІЦІЇ

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Юрій КОРОВКІН

Науковий керівник:
доцент, кандидат технічних наук
Андрій ГОЛОВАТИЙ

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**
(підпис)

Львів
2026

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПК - Персональна обчислювальна машина (комп’ютер).

МК - Мікроконтролер - інтегральна схема для керування електронними пристроями.

ЕС - Електрична схема (взаємозв'язок компонентів системи).

АЗ - Апаратне забезпечення (Hardware).

ПЗ - Програмне забезпечення (Software).

ЕСО - Система охолодження з функціями енергозбереження.

ВС - Вбудована (ембеддед) система спеціального призначення.

ЦП - Цифровий пристрій.

САПР - Програмний комплекс для автоматизації проектних робіт.

AVR - Лінійка 8-бітних мікроконтролерів, розроблена компанією Atmel.

FLASH - Тип напівпровідникової пам'яті, що підтримує багаторазове електричне перезаписування.

EEPROM - Енергонезалежна пам'ять, що допускає побайтове стирання та запис даних електричним способом.

ПЗП (ROM) - Пристрій для постійного зберігання даних (тільки для читання).

ОЗП (RAM) - Оперативна пам'ять для тимчасового зберігання даних під час роботи системи.

АЦП (ADC) - Модуль для конвертації аналогового сигналу у цифровий код.

-РКД (LCD) - Екран на основі рідких кристалів.

SPI - Синхронний протокол для швидкої послідовної передачі даних між МК та периферією у повнодуплексному режимі.

I2C - Послідовна шина для обміну даними між мікросхемами за допомогою двох ліній: сигнальної (SDA) та синхронізації (SCL).

АНОТАЦІЯ

Коровкін Ю., Головатий А. (керівник). Розроблення трекера сонячної електростанції для автономних блокпостів поліції. Бакалаврська кваліфікаційна робота. – Львівсьий державний університет внутрішніх справ, Львів, 2026.

У кваліфікаційній роботі створено комплекс апаратних і програмних засобів для трекера сонячної електростанції для автономних блокпостів поліції. Система здійснює автономне позиціонування сонячної панелі електростанції на Сонце, відображаючи отримані дані на символьному РК-дисплеї. Система приймає рішення щодо активації необхідних виконавчих пристроїв для забезпечення точної орієнтації сонячної панелі.

Сонячний трекер оснащений меню, яке дозволяє встановлювати необхідні параметри для автономної роботи. Здійснюючи оперативне налаштування сонячної панелі на Сонце, електростанція досягає максимальної ефективності.

Трекер керується платою ARDUINO UNO, до якої під’єднано цифровий акселерометр ADXL345, годинник реального часу DS1307, цифровий енкодер AS5600, мотори виставлення азимуту і альтитуди та РК-дисплей.

У роботі спроєктовано електричну принципову схему та створено модель автономного трекера в САПР Proteus VSM. Розроблено алгоритм його функціонування та програмний код на мові С в середовищі розробки ARDUINO IDE. Крім того, здійснено моделювання функціонування автономного трекера в Proteus ISIS.

Ключові слова: сонячний трекер, цифровий акселерометр ADXL345, годинник реального часу DS1307, цифровий енкодер AS5600, символьний РК-дисплей, МК AVR, САПР Proteus VSM, мова програмування С, середовище розробки ПЗ для МК AVR ARDUINO IDE.

ABSTRACT

Korovkin Yu., Holovatyi A. (Supervisor). Development of a Solar Power Plant Tracker for Autonomous Police Checkpoints. Bachelor's Thesis. – Lviv State University of Internal Affairs, Lviv, 2026.

This qualification thesis presents the development of a hardware and software complex for a solar tracker designed for autonomous police checkpoints.

The system performs autonomous positioning of the solar panel towards the Sun, displaying the acquired data on a character LCD. The system makes decisions regarding the activation of the necessary actuators to ensure precise orientation of the solar panel.

The solar tracker is equipped with a menu that allows users to set the required parameters for autonomous operation. By performing real-time adjustment of the solar panel towards the Sun, the power plant achieves maximum efficiency.

The tracker is controlled by an ARDUINO UNO board, which is interfaced with an ADXL345 digital accelerometer, a DS1307 real-time clock, an AS5600 digital encoder, azimuth and altitude adjustment motors, and an LCD.

Within the scope of this work, a schematic electrical diagram was designed, and a model of the autonomous tracker was created using Proteus VSM CAD. An operational algorithm and C-language software code were developed within the ARDUINO IDE environment. Furthermore, the functional simulation of the autonomous tracker was conducted in Proteus ISIS.

Keywords: solar tracker, ADXL345 digital accelerometer, DS1307 real-time clock, AS5600 digital encoder, character LCD, AVR microcontroller, Proteus VSM CAD, C programming language, ARDUINO IDE software development environment for AVR microcontrollers.

ЗМІСТ

ВСТУП

Завдяки прогресу в галузі електроніки, зокрема появі доступних мікроконтролерів та сенсорів, стало можливим створення бюджетних автоматизованих систем для моніторингу та управління. Одним із прикладів практичного застосування таких технологій є сонячний трекер, призначений для забезпечення енергонезалежності поліцейських блокпостів. Постійне збільшення попиту на електроенергію змушує розробників зосереджуватися на підвищенні ККД мобільних фотоелектричних станцій, що є пріоритетним завданням при проектуванні сучасних систем керування.

Метою роботи є розроблення апаратного та програмного забезпечення трекера сонячної електростанції для автономних блокпостів поліції, який працює автономно і забезпечує точне позиціонування сонячної панелі на Сонце. При використанні системи стеження за сонцем вироблена потужність може досягати 40-50% від номінальної, на відміну від стаціонарно встановлених панелей, у яких цей показник - 10-15%.

Система автономно обчислює азимут і альтитуду (висоту Сонця над горизонтом) аналізує ці значення та приймає рішення згідно з алгоритмом роботи про ввімкнення відповідних виконавчих механізмів (моторів позиціонування.)

Перевага системи спостереження за сонцем (сонячного трекера) полягає в тому, що розміщені на них сонячні батареї автоматично рухаються за сонцем протягом дня і міняють кут нахилу в залежності від пори року. А це - значно збільшує вироблення електроенергії порівняно з нерухомо закріпленими сонячними панелями. Необхідна інформація відображається на символічному рідкокристалічному дисплеї. Для розроблення апаратного забезпечення трекера сонячної електростанції для автономних блокпостів поліції пропонується використовувати платформу ARDUINO UNO з МК AVR від компанії Microchip (Atmel).

Використання мікропроцесорної техніки дозволяє розробникам ефективніше втілювати складні функціональні алгоритми, роблячи процес проектування дешевшим. Оскільки сучасні мікроконтролери вже мають у своїй структурі необхідні модулі для обробки сигналів та керування, потреба у допоміжних вузлах зводиться до мінімуму. Результатом такого підходу стає створення малогабаритних приладів із покращеними характеристиками енергозбереження.

Аналіз останніх досліджень і публікацій. Питанням застосування мікроконтролерних засобів у розробці цифрової апаратури присвячено численні праці та прикладні проєкти. Ці дослідження створюють теоретичне підґрунтя для подальшого вдосконалення автоматизованих комплексів моніторингу й управління. Варто наголосити, що технології позиціонування сонячних панелей виконавчими механізмами постійно вдосконалюються. У цьому контексті проектування трекера сонячної електростанції для автономних блокпостів поліції на базі архітектури AVR є затребуваним і своєчасним завданням. Викладене вище дозволяє визначити ключову мету та етапи виконання роботи.

Метою роботи є проєктування апаратного і програмного забезпечення трекера сонячної електростанції для автономних блокпостів поліції.

Для досягнення мети необхідно вирішити наступні **завдання**:

- розробити трекер сонячної електростанції на основі наступних даних:
  - Апаратна платформа ARDUINO UNO з МК AVR (ATmega328);
  - цифровий акселерометр ADXL345;
  - цифровий магнетометр AS5600;
  - годинник реального часу DS1307;
  - символьний РКІ 16×2. Модель: WH1602B-YGK-CTK;
- спроєктувати трекер сонячної електростанції в САПР Proteus;
- розробити алгоритм роботи вбудованої системи;

- створити ПЗ для взаємодії з акселерометром ADXL345, з цифровим магнетометром AS5600, з годинником реального часу DS1307, для виведення інформації на символьний РКІ;
- створити основне ПЗ трекера сонячної електростанції згідно алгоритму його роботи;
- дослідити функціонування моделі трекера та проаналізувати отримані дані в середовищі емулятора Proteus ISIS.

Об’єкт дослідження – проєктування трекера сонячної електростанції для автономних блокпостів поліції.

Предмет дослідження – моделі та засоби проєктування трекера сонячної електростанції для автономних блокпостів поліції.

Методи досліджень. Для досягнення поставленої мети та розв'язання визначених завдань було застосовано комплекс наукових методів пізнання. Зокрема, використання бібліометричного підходу дало змогу проаналізувати профільні публікації та простежити еволюцію досліджуваної проблематики. Глибокий теоретичний аналіз предмета роботи забезпечено застосуванням низки філософських методів, серед яких діалектичний, синергетичний, феноменологічний та аксіологічний. Крім того, цілісність дослідження та виконання його прикладних завдань ґрунтуються на використанні загальнонаукового інструментарію: моделювання, аналізу, синтезу, а також індуктивного, дедуктивного та структурного методів.

Структура роботи. Структурно кваліфікаційна робота включає вступну частину, чотири основні розділи, загальні висновки, перелік використаної літератури та додатки. Пояснювальна записка містить 54 сторінки основного тексту, проілюстрованого 36 рисунками та 7 таблицями. Інформаційну базу дослідження складають 11 бібліографічні джерела. Повний обсяг роботи, враховуючи 2 додатки, становить 94 сторінки.

РОЗДІЛ 1
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Системи орієнтації за Сонцем

Для правильного позиціонування сонячної панелі перпендикулярно до сонячних променів (при цьому найбільший коефіцієнт корисної дії панелей), необхідно знати біжучий азимут Сонця та висоту Сонця над горизонтом, які постійно змінюються. Також для обчислень потрібна початкова орієнтація системи (Сонячна панель, поворотний механізм - трекер) відносно сторін світу. (як правило, початково система орієнтується строго на підень. Північ - позаду). Це можна здійснити з допомогою компаса врахувавши магнітне схилення (м-declination).

Щоб відстежувати (рухати за Сонцем сонячну панель) потрібно постійно обчислювати азимут Сонця і його висоту над горизонтом. Для цих обчислень потрібні початкові дані такі як: Координати місця (Locatin) – де встановлена система. (можна взяти з Інтернету.Google map…). Точний час та дата. Магнітне схилення (можна взяти з Інтернету). Залежить від координат та року. Дійсне протягом 3-5 років.

Магнітне схилення (також відоме як магнітна деклінація або магнітна варіація) – це кут між географічною (справжньою) північчю та магнітною північчю у певній точці на Землі. Географічна північ – це напрямок на Північний географічний полюс Землі, а магнітна північ – це напрямок, який вказує стрілка компаса (напрямок на Північний магнітний полюс).

За цими вхідними даними можна обчислити: час сходження Сонця, та азимут сходження; час заходу Сонця, та азимут заходу; час кульмінації Сонця (Сонце в зеніті); біжучий азимут Сонця; біжучу висота Сонця над горизонтом (Altituda), або “кут піднесення”.

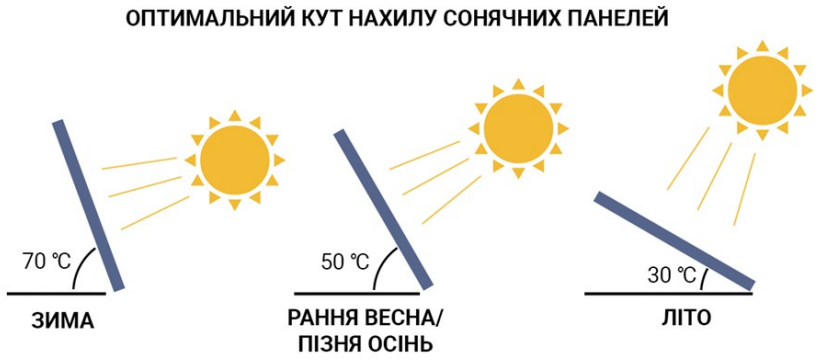
Точність обчислень залежать від точних вхідних даних. Для позиціонування сонячних панелей допустима похибка позиціонування 2-4

градуси. Всі обчислення можна отримувати з online – калькуляторів, якщо є постійно підключений Internet, або робити всі обчислення, з достатньою точністю, автономно. Автономні обчислення є економічно вигідними. Сонячні панелі працюють від сонячного світла, а не від сонячного тепла, тому виробіток електроенергії не знизиться при температурі нижче нуля.

Сонячні панелі будуть працювати і взимку, але їх продуктивність може бути нижча в 2-3 рази. Зниження продуктивності може бути зв'язано з такими факторами, як скорочення світового дня (день взимку коротший, а це значить, що і світла електростанція буде отримувати менше, та від орієнтації панелі відносно джерела енергії (Сонця). Для підвищення продуктивності сонячних електростанцій їх треба правильно орієнтувати за Сонцем.

Тривалість світлового дня коливається у різні пори року. При сприятливих погодних умовах влітку генерується більше електроенергії за рахунок раннього сходу сонця і пізнього заходу за горизонт, взимку робота сонячних електроустановок зменшується через короткий світловий день.

У кожен сезон положення Сонця різне, тому в ідеалі, для кожної пори року підбирається свій кут нахилу. Наприклад, влітку оптимальний кут нахилу складає 30-40 градусів, а взимку — більше 70, в залежності від широти місцевості. Навесні і восени кут нахилу має середнє значення між значенням кута для літа та зими. У північній півкулі потрібно орієнтувати панелі у бік півдня, у південному – у напрямі півночі. Розрахунок кута встановлення сонячних батарей залежить від сезону та географічного положення. Рекомендований кут нахилу рівнозначний широті місцевості.



Для максимальної ефективності сонячні панелі у Львові, як і в усій північній півкулі, слід орієнтувати на південь. Важливо враховувати кут нахилу панелей, який залежить від пори року та широти місцевості. Розрахунок кута нахилу:

- Влітку: широта місцевості мінус 15 градусів.
- Взимку: широта місцевості плюс 15 градусів.
- Наприклад, для Тернополя (49 градусів широти) влітку оптимальний кут буде 49 - 15 = 34 градуси, а взимку 49 + 15 = 64 градуси.

Як було виявлено в численних експериментах та на прикладі реальних об’єктів, кут нахилу суттєво впливає на продуктивність фотомодулів. Обравши не той кут, власник станції може недорахуватись до 20% виробітку. Насправді, для нерухомих сонячних панелей (без сонячного трекера) розрахунок кута зробити надзвичайно просто. Достатньо дізнатися приблизну широту регіону, де розміщується об’єкт. Для визначення оптимального нахилу панелі влітку, віднімемо 15° від значення широти. Взимку ж додаємо 15°. Оберемо широту Львівської області.

Львівська область розміщується між 48°45' та 50°46' північної широти. Оберемо середнє значення 50°. Тому в даному регіоні рекомендується влітку виставляти кут нахилу панелей у 30°, а зимою – 70°. Але є одне але. Виставляти оптимальний кут можна лише в тому випадку, якщо фотомодулі розміщуються на поворотній металоконструкції. Коли ж вони змонтовані на даху або на нерухомій конструкції, відштовхуємось від того, чи хочемо забезпечити максимальний виробіток влітку, чи вибрати середнє значення нахилу (між 70° та 30°) для більш рівномірного показника продуктивності. Обидві стратегії мають свої переваги. В першому випадку, ми максимізуємо значення виробітку станції саме в ті місяці, на які припадає найбільший рівень сонячної інсоляції – це травень, червень, липень та серпень. Модулі, що приймають ультрафіолетові промені, фіксуються в певному положенні, і в більшості випадків (якщо не застосовуються трекери) не можуть рухатися

слідом за сонцем. Тому кут сонячних панелей потрібно ретельно прораховувати.

Якщо система автономна, то змінювати положення слід двічі на рік – у квітні та у вересні. Найкращий випадок, коли не хочеться глибоко вникати в розрахунки та коригувати позицію батарей - зробити монтаж під кутом, який дорівнює широті будинку. Але в такому разі будьте готові до того, що на максимальні показники продуктивності вийти не вдасться.

Кут нахилу безпосередньо впливає на те, як добре енергія Сонця перетворюється сонячними панелями на електричну. Неправильне встановлення веде до втрати 20-30% енергії на рік, якщо попередньо неправильно визначено кут нахилу сонячних панелей. Виробництво електроенергії сонячними електростанціями залежить від кута інсоляції, тобто кута падіння сонячних променів на Землю.

Очевидно, що чим триваліша і вища інтенсивність сонячного освітлення, тим більше енергії дасть кожна панель. Через рух Землі навколо своєї осі та навколо Сонця кут інсоляції протягом року може змінюватись майже на 7%, а продуктивність сонячної електростанції – зменшуватись на 15-20% при неправильному розрахунку кута нахилу.

У літній період Сонце знаходиться вище, отже, кут потрібно робити менше для рівномірного розподілу променів, а взимку, навпаки, панелі слід підняти. Інакше важко зловити максимальну кількість низького зимового сонця. Для збереження оптимальної орієнтації використовуються трекери. Вони допомагають налаштувати кут нахилу сонячних панелей. Розворот групи панелей може здійснюватися: одновісними трекерами – зі зміною лише нахилу; двовісними – з поворотною платформою для зміни нахилу та орієнтації. Електронна частина трекера зчитує інформацію про широту, регіон та місяць і обчислює кут, при якому генерація буде максимальною. Механічна частина повертає панелі. Переваги трекерів у порівнянні з фіксованими системами. Трекери дозволяють збільшити продуктивність електростанції.

Вони сприяють гарній генерації навіть узимку. Датчики трекера вловлюють максимальну кількість сонячних променів, змінюючи положення панелей не лише за сезонами, а й протягом доби.

Робота сонячних трекерів (систем спостереження за сонцем) полягає в тому, що розміщені на них сонячні батареї автоматично рухаються за сонцем протягом дня і міняють кут нахилу в залежності від пори року. А це - значно збільшує вироблення електроенергії порівняно з нерухомо закріпленими сонячними панелями.

Сонячний трекер - установка для отримання максимальної продуктивності від сонячних батарей, шляхом відстеження положення сонця і оптимального орієнтування несучої конструкції. Відомо, що сонячні панелі мають максимальну ефективність, коли їх поверхня спрямована перпендикулярно до сонячних променів протягом усього дня. При використанні системи стеження за сонцем вироблена потужність може досягати 40-50% від номінальної, на відміну від стаціонарно встановлених панелей, у яких цей показник - 10-15%.

1.2. Огляд існуючих підходів до розробки сонячних трекерів

Конструкція сонячного трекера може мати різну конфігурацію і провідний механізм. Пристрій може керуватися різними алгоритмами. Незважаючи на велику кількість відмінностей, трекери поділяються на два основних типи: однокоординатні - здійснюють обертання панелі у напрямку від сходу на захід; двокоординатні - володіють великою рухливістю в двох осях і забезпечують як денне, так і сезонне переміщення за сонцем.

В залежності від способу управління трекери для сонячних батарей діляться на три види:

1. Ручний – управління кутом нахилу і вибір кількості поворотів здійснюється оператором. Продуктивність СЕС в даному випадку залежить від його досвіду і кваліфікації.

2. Пасивний – орієнтація проводиться за заданим алгоритмом для конкретного географічного розташування фотомодулів.
3. Активний – система орієнтує робочу поверхню модуля перпендикулярно до сонця.

Найбільшого поширення набув активний спосіб управління, оскільки є найбільш економічно доцільним і вимагає мінімального обслуговування. Трекер для сонячних панелей в максимальній комплектації складається з: несучою опорною конструкції; вертикально-рухомого механізму; горизонтально-рухомого механізму; електродвигунів; пасової або зубчастої передачі, підшипника; блоку управління автоматикою і актуаторів для орієнтування по координатам; захисту від перевантажень, блискавкозахисту, стабілізаторів; метеостанції - для переорієнтації системи з метою мінімізації пошкоджень від несприятливих погодних умов; системи віддаленого доступу; інвертора - для перетворення постійного струму в змінний, живлення трекера і споживачів.

Така комплектація трекера дорога і не завжди доцільна. Все залежить від його виду і мети застосування. На практиці багато із перерахованих елементів відсутні. Виробники пропонують різні види трекерів. Сонячний трекер SolarSan – комплекс стеження за сонцем, що використовує астрономічний алгоритм. Вбудований GPS-приймач п’ятого покоління. За допомогою SOLARSAN можна орієнтувати сонячні концентратори, колектори або панелі під прямим кутом до сонячного світла. Керування здійснюється за допомогою актюаторів або поворотних приводів в одній або двох площинах і має точність один градус. Модуль має вхід для підключення «анемометра», для захисту від сильного вітру, вхід «датчика граду та снігу» та вхід “фотодатчика” для режиму сну за низької сонячної активності. Підключення флюгера, а також вбудоване безперебійне джерело живлення. Вбудований WiFi-модуль дозволяє конфігурувати трекер через WEB-сторінку по локальній мережі або

через точку доступу. Живлення від 12V до 30V (і версія HV з живленням від 12V до 55V). Захист по струму.

Системи спостереження за сонцем ST1500 полягає в тому, що розміщені на них сонячні батареї автоматично рухаються за сонцем протягом дня і міняють кут нахилу в залежності від пори року. А це - значно збільшує вироблення електроенергії порівняно з нерухомо закріпленими сонячними панелями. Характеристики системи стеження за сонцем ST1500: виробник: IviTec (Україна); розроблений для сонячної панелі до 2000 Вт (6шт); кут повороту 180 градусів; площини обертання - 2 (двовісний трекер (Рух по параболі)); кількість двигунів -1 (рух по параболі); забезпечує до 40% збільшення вироблення потужності; принцип роботи - розрахунок положення сонця за даними GPS; робоча напруга: 12В/24В/36В/48В; робоча температура: від -40 до + 60 °С; розмір поворотною майданчики: 3000х1580х40мм; можлива комплектація датчиком вітру за доп. плату; документація системи стеження за сонцем ST1500.

Реалізувати систему слідкування за сонцем можна двома основними методами, або безпосередньо стежити за рівнем освітленості, на сенсорній основі, або робити обчислення положення сонця і повертати конструкцію згідно розрахованим координатам.

Одним з суттєвих недоліків першого методу є періодичне затемнення сонця хмарами, в такому випадку значення рівню освітленості буде не стабільним, і система буде відповідно нестабільно себе поводити, через це, було обрано другий метод слідкування за сонцем.

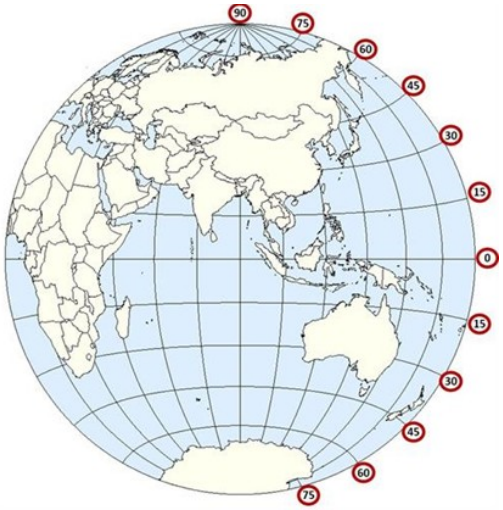
Для повороту конструкції розраховується азимут та висота сонця над горизонтом за актуальним географічним положенням та часом спостереження. Для отримання актуальної дати та часу використовується модуль годинника реального часу на базі мікросхеми DS1307.

Розрахунок положення сонця можна виконати за допомогою бібліотеки “SunPosition”, також використати додаткову бібліотеку “UnixTime” для

конвертації дати та часу у необхідний формат. Для безперервного позиціонування сонячної панелі перпендикулярно до сонячних променів протягом доби (дня) потрібно постійно обчислювати азимут і висоту Сонця над горизонтом (h): (Кут між Сонцем та горизонтом) для даного місця і поточного місцевого часу. Отже небхідні дані для обчислення насамперед є координати місця (широта і довгота де встановлена сонячна панель).

Широта і довгота – це географічні координати, які використовуються для визначення положення точки на земній поверхні. Широта вимірюється у градусах від екватора (0°) до полюсів (90° північної або південної широти). Довгота вимірюється у градусах від нульового меридіана (Гринвіцький меридіан) – від 0° до 180° на схід або на захід. Географічна широта – величина дуги паралелі в градусах від екватора до певної точки. (Кут під яким видно Полярну зорю від лінії горизонту – це і є географічна широта).

Визначається за паралелями, може бути північною (якщо знаходиться вище від екватора) або південною (якщо знаходиться нижче від екватора), змінюються від 0° до 90°.



Всі об'єкти, які знаходяться в Північній півкулі мають північну широту (пн. ш.). Всі об'єкти Південної півкулі мають південну широту (пд. ш.).

Географічна довгота – це відстань в градусах від початкового меридіану до певного пункту. Визначається за меридіанами, може бути західною (якщо об’єкт знаходиться по ліву сторону від 0 меридіану) або східною (якщо об’єкт

знаходиться по праву сторону від 0 меридіану), змінюються від 0° до 180°. Щоб знайти координати місця можна скористатися картами Google. Формат виводу інформації (широта, довгота). Приклади підтримуваних форматів: десяткові градуси: 41.40338, 2.17403; градуси, мінути й секунди: 41°24'12.2"N 2°10'26.5"E; градуси й десяткові мінути: 41 24.2028, 2 10.4418. Важливими параметрами є дата і місцевий час.

Місцевий час – це час на певному меридіані в певний момент, який визначається положенням Сонця. Щоб обчислити місцевий час, потрібно знати різницю між довготами двох місць і врахувати, що кожні 15 градусів довготи відповідають одній годині різниці в часі. За кожні 15° різниці в довготі, час відрізняється на одну годину. Якщо пункт знаходиться на схід від відомого місця, додайте різницю в часі; якщо на захід, відніміть.

Різниця в місцевому часі між двома пунктами становить 4 хвилини на кожен градус різниці в довготі. Формула для обчислення різниці в місцевому часі виглядає так: Різниця в часі (в хвилинах) = Різниця в довготі (в градусах) * 4 хвилини/градус.

Всесвітній координований час (UTC) є місцевим часом на нульовому меридіані, який проходить через Гринвіч, Лондон, Велика Британія. Цей меридіан був міжнародно визнаний як початковий меридіан (Prime Meridian) у 1884 році на Міжнародній меридіанній конференції. Усі інші часові пояси та довготи відраховуються від нього. Отже, якщо ви перебуваєте безпосередньо на довготі 0°, ваш місцевий час буде відповідати UTC.

Київський час більший від UTC на 2 години взимку та на 3 – улітку. “Час кульмінації” Сонця, або його істинний сонячний полудень, це момент, коли Сонце досягає найвищої точки на небі для певної локації. Цей час залежить від довготи місця та рівняння часу. Для обчислення кута піднесення Сонця, або інакше висоти Сонця над горизонтом скористаємося формулою (1). В контексті астрономії та географії терміни “висота Сонця над горизонтом” і

“кут піднесення Сонця”, а також altitude є, по суті, синонімами. Вони всі позначають одне й те саме.

$$\sin(h) = \sin(\varphi) \cdot \sin(\delta) + \cos(\varphi) \cdot \cos(\delta) \cdot \cos(H)$$

$$\sin(h)=\sin(\phi)*\sin(\delta)+\cos(\phi)*\cos(\delta)*\cos(H) \quad (1.1)$$

де h – кут піднесення Сонця, який ми шукаємо; ϕ (фі) – географічна широта місця спостереження; δ (дельта) – схилення Сонця для поточної дати; H – годинний кут Сонця; схилення Сонця (δ): (δ) – це кут між лінією, що з’єднує центр Сонця з центром Землі, і площиною екватора Землі. Воно змінюється протягом року через нахил осі Землі (приблизно 23.45°) відносно площини її орбіти навколо Сонця. Обчислити схилення Сонця можна за допомогою кількох формул, які різняться за точністю та складністю. *Приблизна формула (достатня для багатьох практичних цілей)*. Ця формула є однією з найпоширеніших і досить точних для більшості застосувань, таких як розрахунки для сонячних панелей або загальні астрономічні спостереження.

$$\delta = 23{,}45° \cdot \sin\left(\frac{360°}{365} \cdot (284 + n)\right) \quad (1.1)$$

Де δ – схилення Сонця в градусах; 23,45° - максимальний кут схилення (нахил осі Землі відносно площини орбіти): n – порядковий номер дня року починаючи з 1 січня (n=1) (для 1 лютого n=31+1=32) (для 1 березня n=31+28+1=60) – якщо рік не високосний інакше n=61. Годинний кут Сонця (H): залежить від точного часу та довготи. Годинний кут відраховується від меридіана спостерігача. Він дорівнює 0° в момент проходження Сонця через меридіан (сонячний полудень). На схід від меридіана він негативний, на захід – позитивний. Змінюється зі швидкістю 15° за годину. Годинний кут (H) – це кут між небесним меридіаном спостерігача та меридіаном, на якому знаходиться світило. Він є мірою часу, що минув з моменту кульмінації світила над меридіаном спостерігача. Саме годинний кут пов’язує азимут і висоту. Годинний кут у південь дорівнює нулю (h=0), вважається від’ємним до півдня ($h < 0$) і додатним – після півдня (h>0). При обертанні Землі годинний кут h

протягом доби змінюється, але кут схилення δ в інженерних розрахунках вважається незмінним.

Визначити місцевий зоряний час (LST), або годинний кут Сонця (H) для даного часу та довготи. Це вимагає перетворення UTC на місцевий час, врахування рівняння часу та інших поправок. Для точного розрахунку годинного кута вам потрібні астрономічні формули або онлайн-калькулятор, оскільки потрібно враховувати рівняння часу. Рівняння часу – це різниця між справжнім сонячним часом (який показує сонячний годинник, тобто фактичний рух Сонця по небу) та середнім сонячним часом (який показує наш звичайний годинник, що йде рівномірно). Простими словами, рівняння часу показує, наскільки “швидше” або “повільніше” Сонце рухається по небу порівняно з ідеальним, усередненим рухом, який лежить в основі наших годинників. Ця різниця може сягати до +/-16 хвилин протягом року. Ця різниця виникає через дві основні астрономічні причини:

Еліптична орбіта Землі навколо Сонця. Земля рухається навколо Сонця не по ідеальному колу, а по еліпсу. Згідно із законами Кеплера, коли Земля ближче до Сонця (у перигелії, що відбувається на початку січня), вона рухається швидше. Коли вона далі від Сонця (у афелії, на початку липня), вона рухається повільніше. Через цю змінну швидкість, видимий рух Сонця по небу також є нерівномірним.

Нахил осі обертання Землі до площини її орбіти (екліптики). Вісь обертання Землі нахилена приблизно на 23.45° до площини, по якій вона обертається навколо Сонця. Це призводить до того, що видимий шлях Сонця по небесній сфері (екліптика) нахилений до небесного екватора. Ці два ефекти накладаються один на одного, створюючи складну, але передбачувану криву, яка показує, наскільки справжній сонячний час відрізняється від середнього сонячного часу в кожен день року. Наші годинники йдуть за середнім сонячним часом, який є рівномірним і зручним для повсякденного життя.

Справжній сонячний час, за яким йде сонячний годинник, може відрізнятися від годинникового.

Сонячні панелі. Хоча для більшості аматорських установок це не критично, для оптимізації кута нахилу сонячних панелей протягом року врахування рівняння часу може підвищити ефективність. Рівняння часу (η) – це різниця між істинним сонячним часом ($T_{іст}$) та середнім сонячним часом ($T_{сер}$) для одного і того ж моменту часу. Зазвичай його визначають як: η=Тіст−Тсер (в деяких джерелах може бути навпаки, Тсер−Тіст). Ця величина вимірюється в хвилинах або секундах. Якщо η позитивне, це означає, що Сонце "йде" швидше за годинник (сонячний годинник випереджає механічний). Якщо η негативне – навпаки, Сонце відстає.

Для практичних розрахунків часто використовують емпіричні (наближені) формули, які дають досить точні результати. Однією з таких поширених формул є:

$$\eta = 9{,}87 \cdot \sin(2B°) - 7{,}67 \cdot \sin(B° + 78{,}7°) \qquad (1.2)$$

де η – рівняння часу в хвилинах; B – кут, що залежить від порядкового номера дня року: $B = \frac{360}{365}(N - 81)$ (у градусах). N – порядковий номер дня року, починаючи з 1 січня (N=1). Рівняння часу η показує різницю між справжнім сонячним часом (тобто моментом, коли Сонце перетинає місцевий меридіан) і середнім сонячним часом (який показує годинник). Якщо рівняння часу -1 година, це означає, що справжній сонячний полудень (момент, коли Сонце найвище в небі) настане на 1 годину пізніше, ніж показує ваш середній місцевий час. Отже, якщо полудень за вашим годинником (середній сонячний час) настав о 12:00, а справжній сонячний полудень настане на 1 годину пізніше, то справжній сонячний полудень ще не настав. Він настане приблизно о 13:00 за вашим місцевим часом. Для встановлення кута нахилу сонячної панелі до Сонця – ці розрахунки виконувати кожні 4 хв. (за 4 хв Сонце переміщується по азимуту на 1°). Обчислимо Азимут для конкретної дати часу, географічних координат. Попередньо з допомогою компаса встановимо

сонячну панель лицевою стороною строго на географічний південь. Знайдемо з допомогою компаса магнітний північний полюс. Врахуємо поправки (магнітне схилення).

Азимут – це кут, який відраховується за годинниковою стрілкою від певного опорного напрямку (зазвичай північного) до напрямку на заданий об’єкт або точку. Ось ключові моменти щодо азимуту: Вимірювання: Азимут вимірюється в градусах від 0° до 360°. Опорний напрямок: Зазвичай як опорний напрямок використовується північ. Це може бути: Магнітна північ: напрямок, який вказує стрілка компаса. Географічна (справжня) північ: напрямок на географічний Північний полюс. Відлік завжди ведеться за годинниковою стрілкою. Північ=0° або 360°. Схід=90°. Південь=180°. Захід=270°.

Визначення азимуту за допомогою компаса. Розмістимо компас горизонтально, щоб стрілка вільно оберталася. Дочекаємося, поки стрілка стабілізується і вкаже на магнітну північ (зазвичай червоний кінець стрілки). Це наш 0° (або 360°). Врахуємо Магнітне схилення (девіація): Магнітна північ (на яку вказує компас) рідко збігається зі справжньою географічною північчю. Різниця називається магнітним схиленням. Для Львова магнітне схилення становить приблизно +6° (станом на 2026 рік, тобто магнітна північ відхилена на 6° на схід від географічної). Щоб отримати справжній азимут, потрібно скоригувати магнітний азимут на величину магнітного схилення (додати або відняти, залежно від знаку схилення). Якщо схилення східне (як у Львові), його потрібно додати до показань компаса. Якщо західне – відняти.

Магнітне схилення (також відоме як магнітна деклінація або магнітна варіація) – це кут між географічною (справжньою) північчю та магнітною північчю у певній точці на Землі. Географічна північ – це напрямок на Північний географічний полюс Землі, а магнітна північ – це напрямок, який вказує стрілка компаса (напрямок на Північний магнітний полюс). Магнітне поле Землі є надзвичайно складним і постійно змінюється.

Магнітне поле Землі не є однорідним. Воно має локальні аномалії, викликані геологічними особливостями (наприклад, покладами магнітних руд) у земній корі. Магнітне поле Землі також зазнає незначних добових і річних коливань через вплив Сонця та Місяця. Найважливішим є те, що магнітне поле Землі змінюється протягом десятиліть і століть через процеси у зовнішньому ядрі Землі. Це означає, що значення магнітного схилення для однієї і тієї ж точки буде різним у різні роки. Оскільки прямих "формул" для ручного обчислення немає, магнітне схилення визначають такими способами:

Використання онлайн-калькуляторів або програм. Ці калькулятори використовують складні моделі магнітного поля Землі, такі як Всесвітня Магнітна Модель (WMM - World Magnetic Model) або Міжнародна Референсна Геомагнітна Поле (IGRF - International Geomagnetic Reference Field). Ці моделі регулярно оновлюються (зазвичай кожні 5 років) і враховують як глобальні зміни поля, так і його дрейф.

Для обчислення потрібно ввести географічні координати (широту і довготу) місцезнаходження та поточну дату. Приклад калькулятора: magnetic-declination.com, NOAA National Centers for Environmental Information. За період 5 років ця величина може змінитися на 0,5°. Отже ця константа актуальна протягом п'яти років. Якщо магнітне схилення становить приблизно +7.27°, це означає, що магнітна північ (на яку вказує стрілка компаса) знаходиться на 7.27° на схід від істинної (географічної) півночі.

Формула для обчислення азимуту сходу Сонця. Основна формула для обчислення азимуту (A) світила, коли його висота над горизонтом (h) дорівнює 0∘ (що відбувається при сході або заході), виводиться з формул сферичного трикутника.

$$\cos(A) = \frac{\sin(\delta)}{\cos(\varphi)} \tag{1.3}$$

де A – азимут сходу Сонця. Значення A буде відраховуватися від істинної Півночі; δ – Схилення Сонця (деклінація) для заданої дати. Це кут між напрямком на Сонце і площиною небесного екватора. Його значення

змінюється протягом року від −23.45° до +23.45°; ϕ - широта місця спостереження. Обчислення висоти Сонця (альтитуди) (h):

$$\sin(h) = \sin(\varphi) \cdot \sin(\delta) + \cos(\varphi) \cdot \cos(\delta) \cdot \cos(H) \quad (1.4)$$

де ϕ (широта місця); δ (схилення Сонця на вказану дату); H (годинний кут).

За допомогою вище наведених формул можна з достатньою точністю обчислювати азимут і висоту Сонця над горизонтом на вказану дату і час. Отже, трекер не потребує підключення до Internet і може працювати повністю автономно забезпечуючи достатню точність позиціонування.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕКЕРА СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ ДЛЯ АВТОНОМНИХ БЛОКПОСТІВ ПОЛІЦІЇ

2.1. Система автоматизованого проектування і моделювання програмованих пристроїв Proteus VSM

Proteus VSM представляє собою універсальне середовище для автоматизованого проектування (EDA), орієнтоване на роботу з електронними схемами та мікропроцесорною технікою різних архітектур [2]. Цей програмний продукт, розроблений британською компанією Labcenter Electronics, базується на використанні високоточних математичних моделей радіодеталей. Ключова перевага пакета полягає у можливості симуляції складних пристроїв на базі мікроконтролерів, цифрових сигнальних процесорів (DSP) та інших програмованих компонентів.

Інструментарій Proteus VSM дозволяє не лише створювати принципові електричні схеми та тестувати їх за допомогою віртуальної вимірювальної апаратури, а й проводити комплексне налагодження коду МК. Крім того, пакет містить спеціалізований модуль для трасування друкованих плат, що підтримує функцію 3D-візуалізації готового виробу з урахуванням габаритів компонентів.

Програмний комплекс Proteus VSM забезпечує широку підтримку популярних мікропроцесорних архітектур, серед яких ARM7, AVR, PIC, 8051, Motorola та Basic Stamp. Внутрішня база даних системи налічує понад 6000 аналогових і цифрових компонентів, включаючи розгалужену бібліотеку мікроконтролерів. Важливою особливістю САПР є сумісність із більшістю сучасних компіляторів та асемблерів, що дозволяє моделювати складні багатопроцесорні системи. Потужностей Proteus цілком достатньо для

повного налагодження алгоритмів керування. Функціонально пакет розділений на два ключові модулі:

1) ISIS – графічний редактор для побудови схем та проведення інтерактивної симуляції.
2) ARES – інструмент для проектування топології друкованих плат, що підтримує автоматичне розміщення компонентів та трасування доріжок на основі створеної в ISIS схеми.

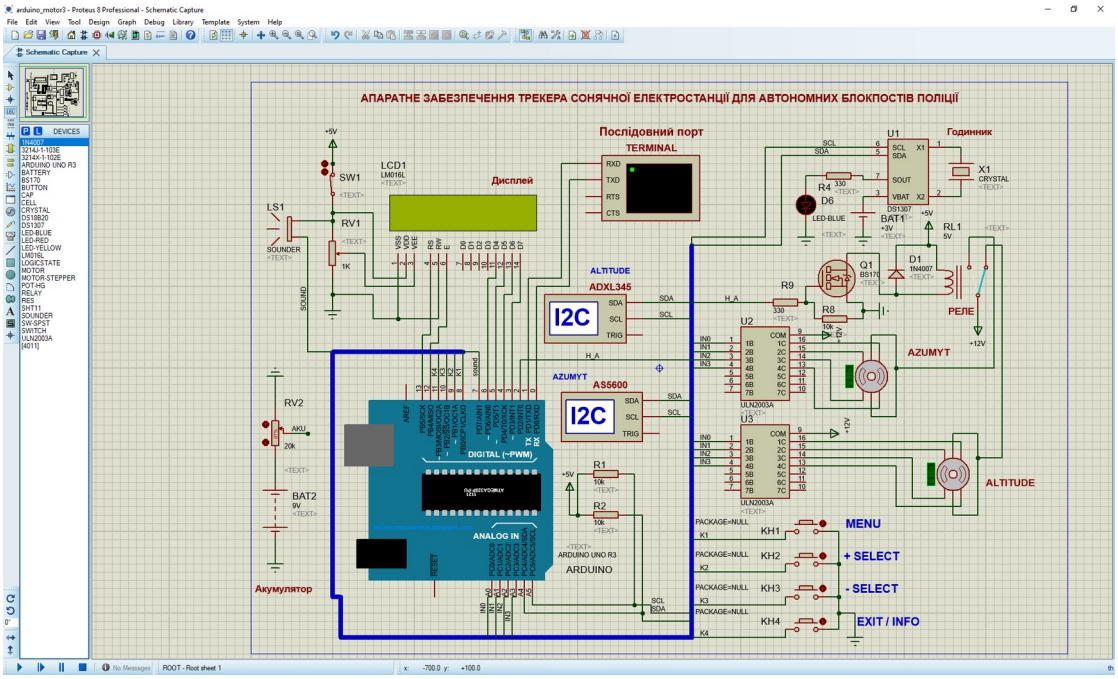


Рис. 2.1. Середовище Proteus ISIS

Для взаємодії віртуальної моделі з реальним обладнанням у Proteus передбачено спеціалізовані компоненти: COMPIM забезпечує зв’язок із фізичним COM-портом комп’ютера, а USBCONN дозволяє під’єднуватися до USB-інтерфейсу. Система демонструє високу сумісність із провідними середовищами розробки (IDE) та компіляторами для різних мікропроцесорних архітектур. Для AVR: підтримуються CodeVisionAVR, WinAVR (з компілятором GCC) та ICC. Для ARM та 8051: інтегроване середовище Keil. Спеціалізовані засоби: HiTECH (для PIC та 8051) та ICC (для ARM7 і msp430).

2.2. Мова С та інструментарій розробки ПЗ Arduino IDE для вбудованих систем на МК AVR

Arduino IDE представляє собою інтегрований комплекс інструментів, що поєднує в собі середовище розробки та гнучку платформу, призначену для оперативного створення програмного коду для різноманітних електронних систем.

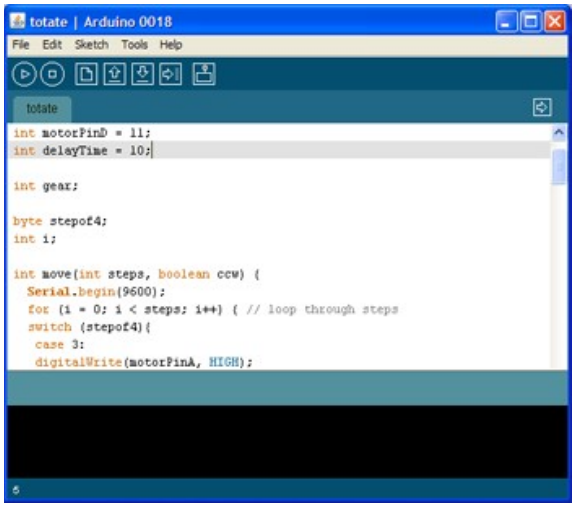


Рис. 2.2. Arduino IDE (Integrated Development Environment)

Arduino IDE – це спеціалізований інструментарій, орієнтований на роботу з однойменними апаратними платформами. Висока популярність цього середовища обумовлена зрозумілим синтаксисом мови, відкритістю програмного коду та наявністю великої кількості готових бібліотек і драйверів. Ключовою особливістю є можливість прошивки контролерів безпосередньо через USB-інтерфейс, це не потребує зовнішніх програматорів.

Інтерфейс програми містить текстовий редактор, термінал для виводу системних повідомлень, консоль та панель керування основними інструментами. Програмний код, розроблений у цьому середовищі, традиційно називають «скетчем», а його базою є мови програмування C та C++. Процес написання коду (скетчу) реалізовано через вбудований текстовий редактор, який підтримує всі стандартні операції: пошук, заміну та редагування фрагментів тексту. Детальна інформація, включаючи розгорнуті звіти про помилки та системні логи, відображається в консолі. Для керування основними процесами розробки передбачена панель інструментів із наступним функціоналом:

- Verify/Compile (Перевірити) – здійснює валідацію коду та його компіляцію.
- Stop (Зупинити) – припиняє роботу монітора послідовного порту.
- New (Новий) – ініціює створення чистого проекту.
- Open (Відкрити) – надає доступ до переліку раніше збережених робіт.
- Save (Зберегти) – фіксує поточні зміни у файлі.
- Upload (Завантаження) – транслює код у машинну мову та записує його в пам’ять контролера Arduino.
- Serial Monitor (Монітор порту) — активує вікно обміну даними з пристроєм.

Додатковий інструментарій згруповано в тематичних розділах меню (*File, Edit, Sketch, Tools, Help*), склад яких може змінюватися залежно від поточного стану роботи програми. Опис основних функцій меню Arduino IDE.

Розділ “Edit” (Редагування). Ця вкладка містить інструменти для експорту коду в різні формати: Copy for Forum: копіює програмний текст у форматі, оптимізованому для публікації в обговореннях на тематичних форумах. Copy as HTML: генерує HTML-еквівалент коду для його коректного відображення на сайтах чи у блогах.

Розділ “Sketch” (Скетч). Тут зібрані команди для управління файлами проекту та залежностями: Include Library: дозволяє інтегрувати необхідну бібліотеку, автоматично додаючи директиву #include до вихідного тексту. Show Sketch Folder: надає швидкий доступ до папки на жорсткому диску, де зберігається поточний проект. Add File...: додає зовнішній файл до скетчу, відкриваючи його у новій вкладці робочого простору.

Розділ “Tools” (Інструменти). Сервісна вкладка для конфігурації обладнання та оптимізації коду: Auto Format: виконує автоматичне вирівнювання структури коду, зокрема розставляє відступи та перевіряє парність дужок для покращення читабельності. Board: дозволяє вказати

конкретну модель плати Arduino, яка використовується в проекті. Serial Port: відображає перелік доступних COM-портів (фізичних та віртуальних). Список оновлюється динамічно при кожному зверненні до пункту меню. Burn Bootloader: призначений для запису завантажувача безпосередньо в пам’ять мікроконтролера. Перед початком процедури важливо переконатися в правильності вибору моделі плати та порту програматора (наприклад, для AVR ISP).

Розділ “File” (Файл). Основним елементом тут є Sketchbook (Блокнот) — це стандартний каталог, де за замовчуванням зберігаються всі проекти. Доступ до них здійснюється через меню або кнопку “Open”.

Arduino IDE підтримує багатозадачність, дозволяючи тримати відкритими кілька проектів одночасно в окремих вкладках. Платформа працює з різними типами вихідних файлів: Власне скетчі Arduino (зазвичай мають розширення .ino, раніше – без розширення). Класичні файли мов програмування C (.c) та C++ (.cpp). Заголовні файли бібліотек (.h).

Процедура прошивки контролера Arduino. Перш ніж ініціювати передачу скетчу, необхідно сконфігурувати параметри зв’язку у вкладках Tools>Board (вибір моделі) та Tools > Serial Port (вибір порту).

У середовищі Windows інтерфейси зазвичай ідентифікуються як COM-порти: системні порти (COM1, COM2) або віртуальні порти для USB-з'єднання (наприклад, COM4, COM7 та інші). Після коректного налаштування слід активувати процес запису через кнопку на панелі керування або через пункт меню Sketch>Upload. Під час транслювання коду на платі Arduino Uno активність обміну даними візуалізується миготінням індикаторів RX та TX.
Ключовим елементом системи є Bootloader (завантажувач) – компактне ПЗ, попередньо записане в пам'ять мікроконтролера. Його наявність дозволяє програмувати пристрій безпосередньо через USB, уникаючи потреби у зовнішніх програматорах. Завантажувач активується в момент рестарту або під час ініціалізації прошивки, про що сигналізує вбудований світлодіод.

Бібліотечне забезпечення. Призначення: надання додаткових інструментів для обробки сигналів та керування залізом. Інсталяція: додавання через меню Include Library або ручне розміщення файлів у каталозі libraries. Оптимізація: підключені компоненти споживають ресурси Flash-пам'яті, що вимагає уважного ставлення до їх переліку в коді.

Послідовний інтерфейс (Serial Monitor). Для спостереження за процесом виконання програми в реальному часі використовується вікно моніторингу порту. Воно дозволяє не лише зчитувати вихідні дані з Arduino, а й надсилати зворотні команди через консольний ввід.

Конфігурація параметрів середовища здійснюється через розділ Preferences. Налаштування, що не доступні безпосередньо в інтерфейсі цього вікна, зберігаються у спеціальному конфігураційному файлі, повний шлях до якого відображається внизу вікна налаштувань. Кожна апаратна платформа Arduino оснащена набором контактів (пінів), призначених для взаємодії з периферійним обладнанням. Ці виводи є універсальними: вони можуть функціонувати як у цифровому, так і в аналоговому режимах (за умови підтримки АЦП).

Для визначення напрямку передачі даних (вхід чи вихід) використовується стандартна функція pinMode(). Важливо враховувати, що цифрові порти можуть перебувати у стані високого імпедансу (Z-стан), що фактично ізолює вхід від внутрішньої шини даних. Щоб уникнути помилкових спрацювань через «плаваючий» потенціал на входах, до яких не підключено зовнішній сигнал, необхідно програмно або апаратно задати стабільний рівень напруги. Для цього застосовують підтягувальні резистори, що з’єднують вхід із лінією живлення (+5 В) або загальним проводом (GND).

Слід пам'ятати про обмеження навантажувальної здатності: максимальний струм одного виводу становить 40 мА. Цього ресурсу недостатньо для прямого живлення двигунів чи потужних реле. Недотримання

цих лімітів або коротке замикання на лінії може призвести до незворотного виходу мікроконтролера з ладу.

Використання широтно-імпульсної модуляції (ШІМ). Принцип роботи ШІМ полягає у формуванні на цифровому виході послідовності прямокутних імпульсів із різним співвідношенням високого та низького рівнів. Це дозволяє імітувати зміну вихідної напруги в діапазоні від 0 до 5 В. Регулювання середнього значення напруги здійснюється шляхом варіації тривалості (ширини) імпульсу при незмінній частоті. У середовищі розробки Arduino для цього використовується функція analogWrite(), де керуюче значення задається в межах від 0 до 255. Наприклад: виклик analogWrite(255) відповідає безперервному сигналу з напругою 5 В (100% коефіцієнт заповнення); команда analogWrite(127) формує сигнал зі шпаруватістю 50%, що еквівалентно середній напрузі 2,5 В.

Характеристики аналогових інтерфейсів та програмна структура. Мікроконтролери сімейства ATmega оснащені вбудованим аналого-цифровим перетворювачем (АЦП) на шість каналів. Завдяки 10-бітній розрядності, вхідний аналоговий сигнал трансформується у цифровий діапазон значень від 0 до 1023. Важливою особливістю аналогових входів є їхня універсальність: за потреби вони можуть функціонувати як стандартні цифрові порти (з нумерацією від 14 до 19). Наприклад, для активації виходу на 14-му піні використовуються типові команди C++: pinMode(14, OUTPUT); // визначення піна як виходу digitalWrite(14, HIGH); // встановлення високого логічного рівня.

Архітектура програмного коду. Перед процесом збірки код автоматично трансформується у повноцінну програму на мові C/C++, після чого обробляється компілятором AVR-GCC. Дане середовище розробки оптимізоване виключно для взаємодії з апаратною системою Arduino.

Апаратні платформи Arduino. Лінійка пристроїв Arduino представлена різними модифікаціями, кожна з яких базується на певному типі

мікроконтролера. Найбільш поширені версії, такі як Uno та Duemilanove, використовують обчислювальну потужність ATmega328.

Для масштабних проектів, що потребують значних ресурсів, розроблено Arduino Mega2560, серцем якої є потужний мікроконтролер ATmega2560. Вибір конкретної моделі є критичним етапом проектування, оскільки він визначає ключові параметри системи: тактову частоту центрального процесора; швидкість обміну інформацією; обсяг доступної пам'яті для розміщення програмного коду (скетчу). На Рис.2.3 зображено апаратну платформу Arduino UNO.

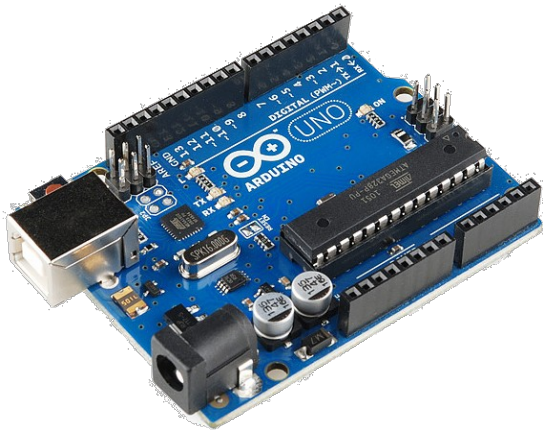
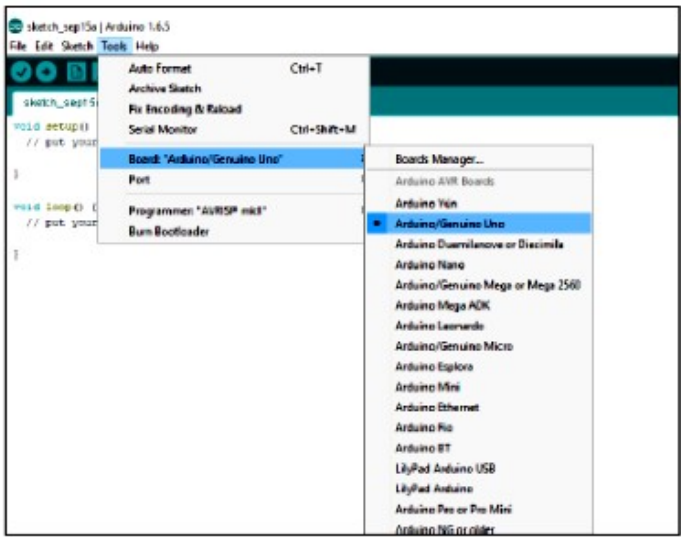
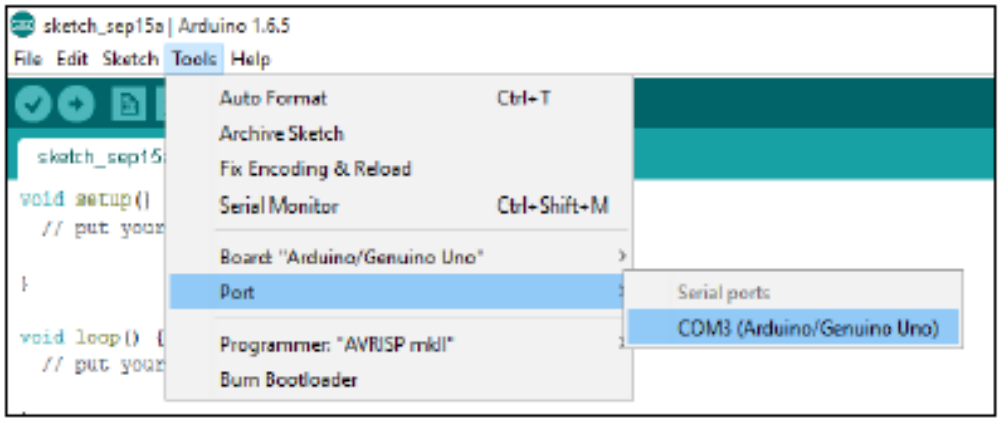


Рис. 2.3. Апаратна платформа Arduino UNO

Для під'єднання ARDUINO UNO до ПК потрібно вибрати саме платформу ARDUINO UNO (Tools>Board>Arduino> Genuino Uno) див. Рисунок.



Далі вибрати порт через який буде здійснено підключення (Tools>Port).



Для завантаження скетчу в мікроконтролер ATmega328p на плату ARDUINO UNO треба вибрати (Sketch> Upload). При завантаженні коду на платі блимає світло діод.

РОЗДІЛ 3
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ТРЕКЕРА СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ ДЛЯ АВТОНОМНИХ БЛОКПОСТІВ ПОЛІЦІЇ

3.1. Вибір елементної бази для проєктування трекера сонячної електростанції для автономних блокпостів поліції

Елементною базою для проєктування трекера було вибрано:

- апаратну платформу ARDUINO UNO з МК AVR (ATmega328);
- трьохосьовий МЕМС акселерометр ADXL345;
- магнітний енкодер AS5600;
- символьний РКД – дисплей 16x2. Модель: WH1602B-YGK-CTK.

Опис апаратної платформи Arduino UNO з МК ATmega328. В основі проектованого пристрою лежить популярна платформа Arduino Uno, яка базується на мікроконтролері ATmega328P від виробника Atmel. Дане рішення було обране завдяки оптимальному поєднанню доступної вартості, високої продуктивності та достатнього набору технічних інтерфейсів. Чіп ATmega328P належить до сімейства 8-бітних мікроконтролерів із RISC-архітектурою. Він характеризується високою швидкістю виконання операцій та гнучкістю налаштувань. Сумісність із середовищем Arduino IDE значно спрощує процес розробки програмного забезпечення, а ресурс перезапису Flash-пам’яті до 10 000 циклів дозволяє проводити багаторазові налагодження пристрою. Ключовою перевагою архітектури AVR є спадкова сумісність систем команд, що дає змогу легко експортувати програмний код між різними моделями контролерів у межах сімейства. На Рис.3.1 зображено плату Arduino UNO та сигнали на її роз’ємах.

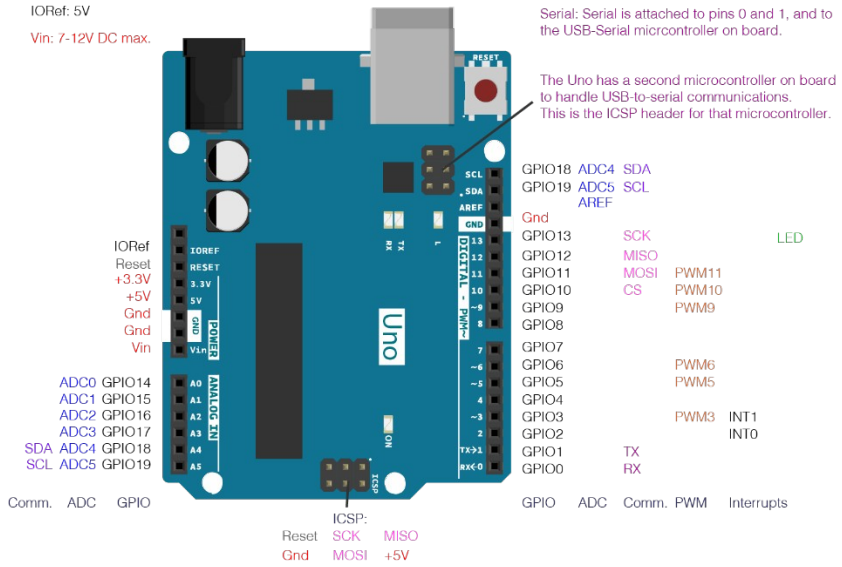


Рис. 3.1. Апаратна платформа Arduino UNO

Мікроконтролер ATmega328P є високоефективним 8-бітним обчислювальним пристроєм із низьким рівнем енергоспоживання, що базується на вдосконаленій RISC-архітектурі. Його ключові параметри та функціональні можливості включають:

Обчислювальні можливості:

- Продуктивність: Завдяки системі зі 131 команди (більшість з яких виконується за один такт), МК забезпечує швидкодію до 16 MIPS при тактовій частоті 16 МГц.
- Регістрова пам’ять: Містить 32 робочих регістри загального призначення, інтегрованих безпосередньо в процесорне ядро.
- Flash-пам’ять програм: Об’єм складає 32 Кбайт із ресурсом до 10 000 циклів перезапису.
- Енергонезалежна пам’ять (EEPROM): 1 Кбайт для зберігання даних із можливістю перезапису до 100 000 разів.
- Оперативна пам’ять (SRAM): Внутрішній статичний накопичувач об’ємом 2 Кбайти.

Периферійні модулі та інтерфейси

- Таймери та лічильники: Система включає два 8-бітних та один 16-бітний програмовані таймери, а також сторожовий таймер (Watchdog).

- Аналоговий блок: 10-розрядний АЦП на 8 каналів та інтегрований аналоговий компаратор.
- ШІМ-генерація: Передбачено 6 каналів для формування широтно-імпульсної модуляції.
- Комунікації: Підтримка апаратних інтерфейсів USART, SPI, а також можливість програмування та налагодження через JTAG.

Живлення та режими роботи

- Енергозбереження: Підтримується шість режимів сну, включаючи Idle, Power-down, Standby та режим мінімізації шумів для точних вимірювань АЦП.
- Стабільність: Наявність схеми скидання при просіданні напруги (Brown-out Reset) та вбудованого каліброваного RC-генератора.
- Електричні параметри: Робочий діапазон напруги становить від 2,7 В до 5,5 В при частоті до 16 МГц.

Конструктивне виконання

- Порти вводу-виводу: 23 програмовані лінії введення/виведення (GPIO).
- Корпуси: Мікросхема випускається у 28-вивідному виконанні PDIP або компактному 32-вивідному корпусі TQFP.

Розташування та призначення контактів для різних типів корпусів наведено нижче.

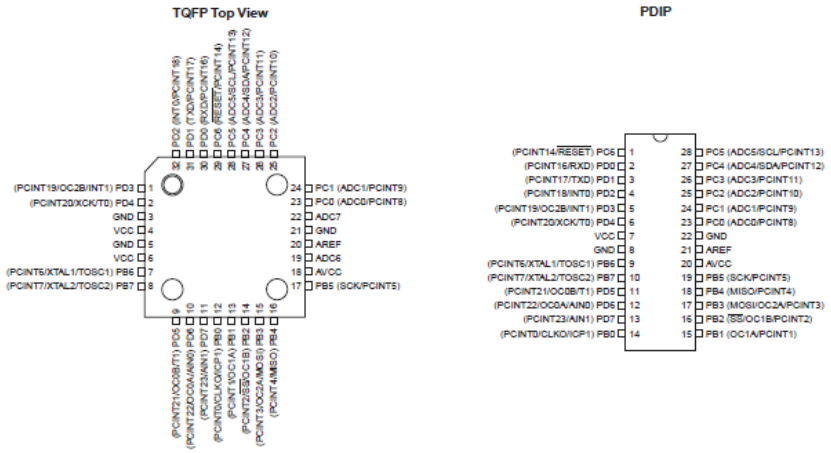


Рис. 3.2 Призначення виводів в корпусах TQFP і PDIP МК AVR ATmega328P

На Рис.3.3 зображено блок-схему МК ATmega328P.

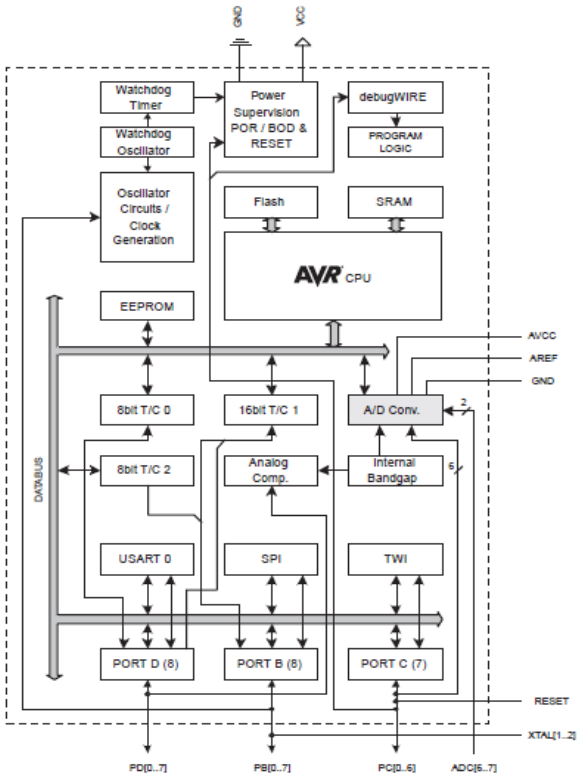


Рис. 3.3. Блок-схема ATmega328P

Для оновлення вмісту FLASH-пам’яті передбачено кілька способів: використання послідовного SPI-інтерфейсу за допомогою типового зовнішнього програматора або застосування спеціалізованого сервісного ПЗ – завантажувача (bootloader).

Мікроконтролер ATmega328P є центральним обчислювальним компонентом популярної платформи Arduino Uno. Повна електрична принципова схема цього пристрою, що демонструє обв'язку та підключення МК, наведена на Рис. 3.4.

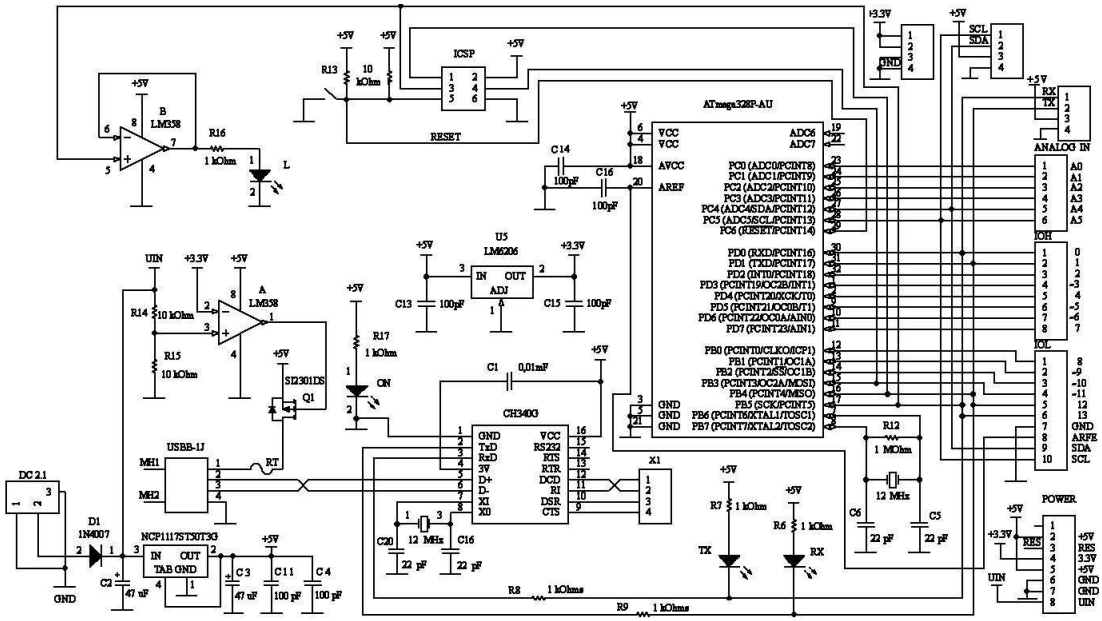


Рис. 3.4. Принципова електрична схема Arduino UNO

Опис акселерометра ADXL345. ADXL345 є триосьовим пристроєм, виконаним за технологією мікроелектромеханічних систем (MEMS). Його функціонал дозволяє моніторити як статичне прискорення (силу тяжіння), так і динамічні впливи, зумовлені вібраціями, ударами або зміною швидкості руху. Прискорення – це зміна швидкості руху тіла за одиницю часу, вимірюється в метрах за секунду в квадраті (m/s^2), або в G-силах (G-сила рівна 9.8 m/s^2). Основними сферами застосування таких сенсорів є моніторинг вібраційних процесів, визначення просторової орієнтації механізмів та аналіз кінематики рухомих тіл.

Сучасний ринок електронної компонентної бази пропонує велику кількість МЕМС-акселерометрів, які класифікують за кількістю осей вимірювання (від однієї до трьох). Найбільш універсальними є триосьові моделі, що дозволяють отримувати дані за векторами X, Y та Z. При виборі конкретного пристрою розробники зазвичай керуються такими критеріями, як роздільна здатність (чутливість), геометричні розміри, діапазон напруги живлення та вартість.

Для практичної реалізації пристрою було обрано триосьовий акселерометр ADXL345. Дана модель є оптимальним рішенням з огляду на її

цінову доступність та відповідність технічних характеристик вимогам проекту. Параметри сенсора забезпечують необхідну точність вимірювань при збереженні низької собівартості апаратної частини.

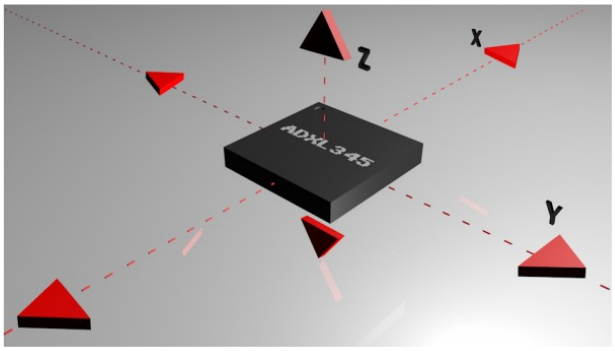


Рис. 3.5. Осі вимірювання трьохосьового акселерометра ADXL345

В основі роботи сучасних МЕМС-сенсорів лежить система мікроскопічних конденсаторних обкладок, виконаних у формі гребінчастих структур. Конструктивно датчик складається з нерухомих елементів та рухомої маси, яка підвішена на пружних демпферах (пружинах). Під впливом сил інерції рухома частина зміщується відносно стаціонарних пластин. Ця деформація спричиняє варіацію електричної ємності між обкладками. Шляхом вимірювання цих змін вбудована електроніка визначає амплітуду та вектор діючого прискорення.

Ключові параметри акселерометрів. Більшість сучасних датчиків підтримують налаштування меж вимірювання, які можуть варіюватися від ±1g до ±250g. Існує зворотна залежність між робочим діапазоном та роздільною здатністю датчика. Тобто при вужчому діапазоні забезпечується вища точність і чутливість до мінімальних відхилень.

Сенсори з низьким порогом (наприклад, ±2g) ідеально підходять для побутової електроніки та крокомірів, тоді як пристрої з широким діапазоном (до 250g) використовуються у спеціалізованій техніці, наприклад, для моніторингу параметрів польоту ракет.

Для ознайомлення з архітектурою пристрою на Рис. 3.6 представлено функціональну схему акселерометра ADXL345.

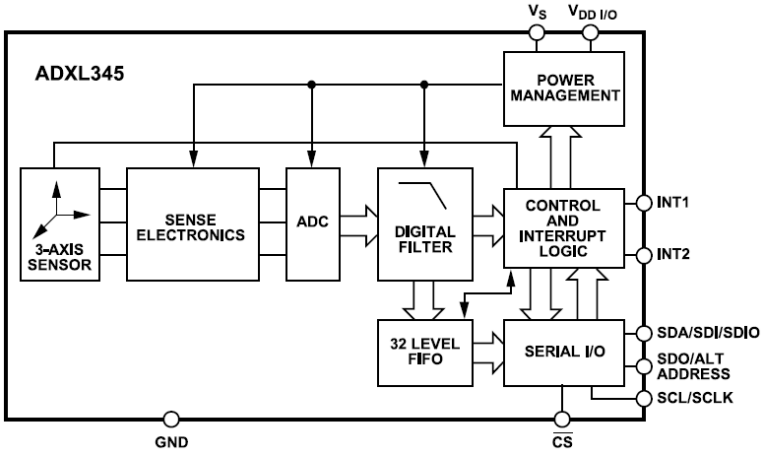


Рис. 3.6. Функціональна схема акселерометра ADXL345

ADXL345 – це компактний триосьовий сенсор із 13-бітною роздільною здатністю. Користувач може самостійно налаштувати межі вимірювань, обравши один із варіантів: ±2g, ±4g, ±8g або ±16g. Результати обробки сигналів передаються у 16-бітному форматі через послідовні протоколи обміну даними I2C або SPI. Завдяки робочій смузі пропускання від 0,05 Гц до 1600 Гц, акселерометр ефективно реєструє як статичне гравітаційне прискорення, так і низькочастотні вібраційні коливання. Під смугою пропускання розуміють здатність сенсора коректно реєструвати динамічні зміни прискорення, що відбуваються з високою частотою (наприклад, вібраційні процеси до 1200 Гц). На ринку існують спеціалізовані високочастотні мікросхеми, розраховані на значні перевантаження (від ±70g до ±500g) та робочі частоти до 30 кГц. Однак такі компоненти є дефіцитними та вирізняються високою вартістю, що робить їх використання в побутових приладах недоцільним. Для розробки трекера сонячної електростанції для автономних блокпостів поліції оптимальним рішенням є акселерометр ADXL345.

Технічні параметри давача ADXL345. ADXL345 має наступний набір експлуатаційних та функціональних характеристик:

- Робочий діапазон напруги становить від 2,0 В до 3,6 В.
- Можливість програмного вибору шкал прискорення між ±2g, ±4g, ±8g та ±16g.

- Режим фіксованої точності — 10 біт.
- Режим максимальної деталізації — до 13 біт при налаштуванні ±16g. Це забезпечує стабільну чутливість на рівні 4 мг/LSB незалежно від обраного діапазону. Підтримка цифрових послідовних протоколів передачі даних SPI та I2C.

Наявність двох програмованих виводів для генерації сигналів переривання за певними подіями. Широкий температурний режим експлуатації: від -40°C до +85°C. Висока механічна міцність: здатність витримувати ударні навантаження амплітудою до 10 000g. На Рис. 3.7. зображено піни мікросхеми ADXL345

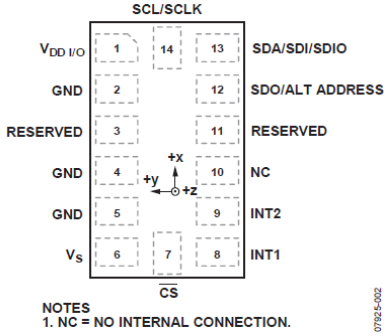


Рис. 3.7. Виводи мікросхеми ADXL345

Призначення цих пінів в Табл.3.1.

Таблиця 3.1. Призначення виводів ADXL345

Вивід	Позначення	Опис
1	$V_{DDI/O}$	Напруга живлення цифрового інтерфейсу
2	**GND**	Земля
3	**RESERVED**	Зарезервовано. Під'єднати до Vs або залишити не під'єднаним
4	**GND**	Земля
5	**GND**	Земля
6	**Vs**	Напруга живлення
7	**CS**	Вибір мікросхеми (лог.0)
8	**INT1**	Вихід переривання 1
9	**INT2**	Вихід переривання 2
10	**NC**	Не використовується (залишити не під'єднаним)
11	**RESERVED**	Зарезервовано. Під'єднати до GND, або залишити не під'єднаним
12	**SDO/ALT ADDRESS**	Вихід послідовної предачі даних(SPI 4-wire) Вибір адреса I2C
13	**SDA/SDI/SDIO**	Послідовна лінія даних (I2C)/Вхід прийому даних (SPI)
14	**SCL/SCLK**	Послідовна лінія тактування. SCL для I2C і SCLK для SPI

Таблиця 3.2 містить інформацію про 31 регістр акселерометра ADXL345

Таблиця 3.2. Регістри ADXL345

Адрес Hex	Адрес Dec	Назва регістра ADXL345	Тип	Значення По-замовчуванню	Призначення
0X00	0	DEVID	R	11100101	ID – пристрою
0X01 ..0X1C	1..28	Reserved	R/W	00000000	Зарезервовано
0X1D	29	THRESH_TAP	R/W	00000000	Чутливість до легкого удару
0X1E	30	OFSX	R/W	00000000	Зміщення осі X
0X1F	31	OFSY	R/W	00000000	Зміщення осі Y
0X20	32	OFSZ	R/W	00000000	Зміщення осі Z
0X21	33	DUR	R/W	00000000	Тривалість поштовху
0X22	34	Latent	R/W	00000000	Затримка поштовху
0X23	35	Window	R/W	00000000	Тимчасове вікно для другого поштовху
0X24	36	THRESH_ACT	R/W	00000000	Поріг активності
0X25	37	THRESH_INACT	R/W	00000000	Поріг неактивності
0X26	38	TIME_INACT	R/W	00000000	Час неактивності
0X27	39	ACT_INACT_CTL	R/W	00000000	Управління осями для визначення активності
0X28	40	THRESH_FF	R/W	00000000	Поріг вільного падіння
0X29	41	TIME_FF	R/W	00000000	Час вільного падіння
0X2A	42	TAP_AXES	R/W	00000000	Управління осями для поштовху/подвійного
0X2B	43	ACT_TAP_STATUS	R	00000000	Джерело поштовху/ подвійного поштовху
0X2C	44	BW_RATE	R/W	00001010	Управління частотою вибірки і режимами живлення
0X2D	45	POWER_CTL	R/W	00000000	Управління енергозбереженням
0X2E	46	INT_ENABLE	R/W	00000000	Управління дозволом переривань
0X2F	47	INT_MAP	R/W	00000000	Призначення виводів переривання
0X30	48	INT_SOURCE	R	00000010	Джерело переривань
0X31	49	DATA_FORMAT	R/W	00000000	Налаштування формату даних
0X32	50	DATAX0	R	00000000	X - вісь DATA 0
0X33	51	DATAX1	R	00000000	X - вісь DATA 1
0X34	52	DATAY0	R	00000000	Y - вісь DATA 0
0X35	53	DATAY1	R	00000000	Y - вісь DATA 1
0X36	54	DATAZ0	R	00000000	Z - вісь DATA 0
0X37	55	DATAZ1	R	00000000	Z - вісь DATA 1
0X38	56	FIFO_CTL	R/W	00000000	Управління FIFO
0X39	57	FIFO_STATUS	R	00000000	Статус FIFO

Споживання ADXL345 електроенергії пропорційно до частоти внутрішнього АЦП (Табл.3.3).

Таблиця 3.3. Електроспоживання ADXL345

Частота дискретизації (Гц)	Полоса пропускання (Гц)	Код	Струм Споживання (μA)
3200	1600	1111	140
1600	800	1110	140
800	400	1101	140
400	200	1100	140
200	100	1011	140
100	50	1010	140
50	25	1001	90
25	12,5	1000	60
12,5	6,25	0111	50
6,25	3,13	0110	45
3,13	1,56	0101	40
1,56	0,78	0100	34
0,78	0,39	0011	23
0,39	0,20	0010	23
0,20	0,10	0001	23
0,10	0,05	0000	23

Інтерфейс ADXL345. Для обміну даними з акселерометром ADXL345 передбачено використання протоколів I2C або SPI. У будь-якій конфігурації пристрій функціонує виключно в режимі веденого вузла (Slave). Вибір конкретного протоколу залежить від необхідної частоти оновлення даних. Для високих частот (1600 Гц та 3200 Гц): Рекомендується застосовувати інтерфейс SPI із тактовою частотою шини не менше 2 МГц. Для середніх частот (800 Гц): Оптимальною є робота через I2C на швидкості 400 кГц. Для нижчих значень частоти дискретизації швидкість передачі можна зменшувати пропорційно. Щоб активувати режим I2C, на контакт вибору кристала (CS) необхідно подати високий логічний рівень (“1”). Ідентифікація пристрою на шині I2C залежить від стану виводу ALT ADDRESS. Основна адреса (ALT ADDRESS = “1”): 7-бітна адреса пристрою – 0x1D. З урахуванням біта напрямку даних (R/W): операція запису здійснюється за адресою 0x3A, а зчитування – за 0x3B. Альтернативна адреса (ALT ADDRESS=”0”): 7-бітна адреса пристрою – 0x53. Повна 8-бітна послідовність (адреса+біт R/W) становить 0xA6 для запису та 0xA7 для отримання даних.

При типовому підключенні за протоколом I2C виводи CS та ALT ADDRESS зазвичай заземлюються (подається “лог. 0”), що відповідає схемі, наведеній на Рис.3.8

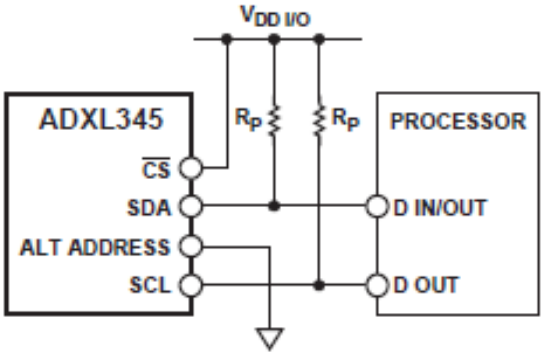


Рис. 3.8. Схема вмикання ADXL345 для роботи по I2C

Для оперативного реагування на зовнішні події акселерометр ADXL345 оснащений двома фізичними лініями переривань – INT1 та INT2. У стандартному режимі роботи активним станом для цих виходів є високий логічний рівень. Розробник має можливість інвертувати цей сигнал, встановивши біт INT_INVERT у регістрі DATA_FORMAT у значення “1”. Така гнучкість дозволяє адаптувати сенсор до вимог входу мікроконтролера.

Особливості керування перериваннями. Активація: сенсор дозволяє одночасно відстежувати декілька подій. Для цього необхідно ініціалізувати відповідні біти в регістрі маскування переривань INT_ENABLE. Маршрутизація: Вибір конкретного фізичного виводу (INT1 або INT2) для кожної події здійснюється за допомогою регістра INT_MAP. Це дає змогу гнучко розподіляти сигнали між входами МК.

Для коректної взаємодії з сенсором необхідно використовувати відповідні адреси регістрів: 1. Ідентифікація джерела переривання: Щоб визначити, яка саме подія (наприклад, падіння чи торкання) викликала сигнал на виході INT, слід звернутися до регістра статусу INT_SOURCE (адреса 0x30). 2. Отримання координат: Результати вимірювань по осях X, Y та Z представлені у вигляді шести 8-бітних регістрів (від DATAX0 до DATAZ1). Процес зчитування рекомендується розпочинати з адреси 0x32 послідовним пакетом. 3. Налаштування параметрів: Конфігурація діапазону вимірювань

(наприклад, ±2g або ±16g) здійснюється через регістр керування DATA_FORMAT.

0x31 DATA_FORMAT (read/write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	X	FULL_RES	JUSTIFY	RANGE	

Керування роздільною здатністю (FULL_RES): Цей параметр визначає точність представлення даних. При встановленні значення «1» активується режим повної роздільної здатності (Full-Resolution), де кількість біт зростає пропорційно діапазону. Якщо біт встановлено у “0”, пристрій працює у стандартному 10-бітному режимі.

Вирівнювання результатів (JUSTIFY): Даний біт регулює розміщення отриманих даних у вихідних регістрах. Значення «0» відповідає вирівнюванню по правому краю (стандарт для математичних операцій), а «1» — по лівому.

Вибір робочої шкали (RANGE): За допомогою комбінації цих бітів задається максимальна межа прискорення, яку здатен зафіксувати сенсор. Детальні варіанти налаштувань для різних значень g наведено у Табл.3.4.

Таблиця 3.4. Діапазон вимірювання ADXL345

D1	D0	g-Range
0	0	±2 g
0	1	±4 g
1	0	±8 g
1	1	±16 g

Найбільш раціональним для функціонування акселерометра є використання 13-бітної роздільної здатності. Для активації цього режиму необхідно встановити біт D3 (FULL_RES) у регістрі DATA_FORMAT у стан логічної одиниці. Перевагою такої конфігурації є незмінна точність вимірювань: незалежно від обраного робочого діапазону (g), ціна молодшого розряду залишається стабільною і становить приблизно 3,9 mg.

Для спрощення процесу розробки доцільно використовувати готові апаратні модулі. Такі плати вже містять мікросхему ADXL345 разом із

необхідною обв’язкою (конденсаторами, стабілізаторами) та виведеним на роз’єм інтерфейсом для зручного з'єднання з мікроконтролером. Зовнішній вигляд та схеми таких модулів представлені на Рис.3.9 та 3.10.

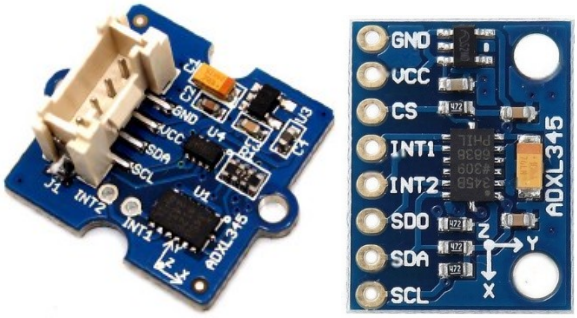


Рис. 3.9. Модулі з ADXL345 акселерометром

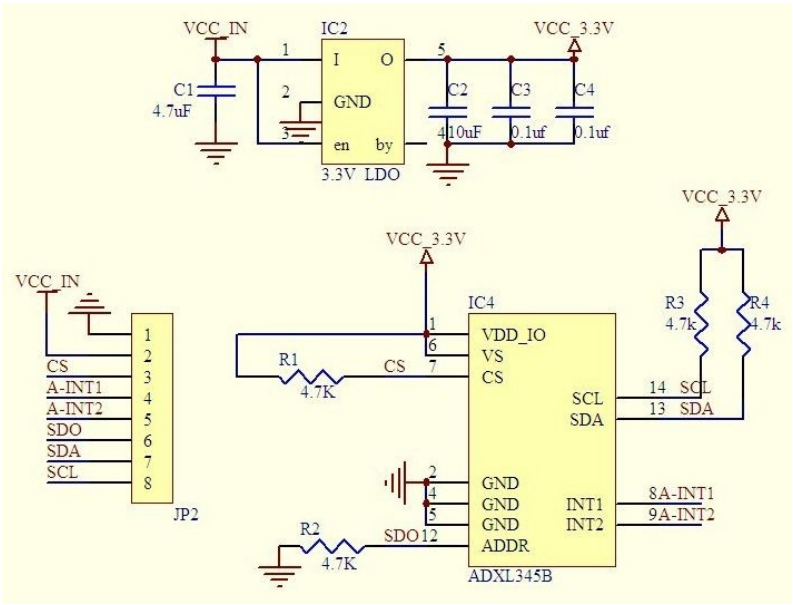


Рис. 3.10. Схема модуля GY-291 з ADXL345

Огляд магнітного сенсора кутового положення AS5600. AS5600 – це простий у програмуванні магнітний роторний датчик положення з 12-бітовим аналоговим або ШІМ-виходом високої роздільної здатності. Ця безконтактна система вимірює абсолютний кут діаметрально намагніченого співвісного магніту. AS5600 розроблений для застосувань безконтактних потенціометрів, а його надійна конструкція усуває вплив будь-яких однорідних зовнішніх блукаючих магнітних полів. Має стандартний промисловий інтерфейс I²C. За замовчуванням вихідий сигнал є в діапазоні від 0 до 360 градусів.

AS5600 також має функцію режиму низького енергоспоживання для автоматичного зменшення споживання енергії. Технічні характеристики: надійність та довговічність; безконтактне вимірювання кута; просте програмування; прості програмовані користувачем початкові та кінцеві позиції через інтерфейс I²C: вихідний сигнал високої роздільної здатності (12-бітна роздільна здатність ЦАП); вибір виходу (аналоговий вихід, ратіометричний до VDD, або цифровий вихід з ШІМ-кодуванням); низьке енергоспоживання (автоматичний перехід у режим низького енергоспоживання); температурний діапазон: -40°C to +125°C.

AS5600 ідеально підходить для безконтактних потенціометрів, безконтактних ручок, педалей, сервоприводів RC та інших рішень для вимірювання кутового положення.

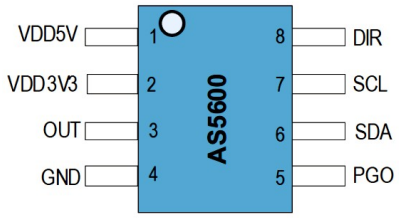


Рис. 3.11. Мікросхема магнітного сенсора AS5600

Призначення виводів AS5600 наведено у Табл.3.5.

Таблиця 3.5. Виводи сенсора AS5600

Pin Number	Name	Type	Description
1	VDD5V	Supply	Positive voltage supply in 5V mode
2	VDD3V3	Supply	Positive voltage supply in 3.3V mode (requires an external 1-µF decoupling capacitor in 5V mode)
3	OUT	Analog/digital output	Analog/PWM output
4	GND	Supply	Ground
5	PGO	Digital input	Program option (internal pull-up, connected to GND = Programming Option B)
6	SDA	Digital input/output	I²C Data
7	SCL	Digital input	I²C Clock
8	DIR	Digital input	Direction polarity (GND = values increase clockwise, VDD = values increase counterclockwise)

Вхідний пін (DIR) задає режим щодо напрямку обертання. Якщо DIR підключений до землі (GND), вихідне значення збільшується при обертанні за

годинниковою стрілкою. Якщо DIR підключений до VDD, вихідне значення збільшується при обертанні проти годинникової стрілки (див. Рисунок 3.12.).

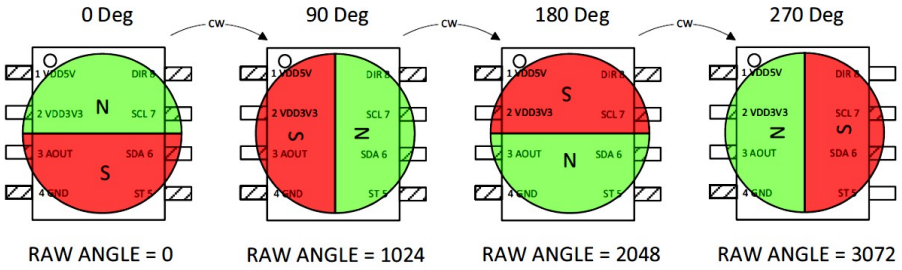


Рис. 3.12. Збільшення вихідного значення AS5600 при DIR=VDD

AS5600 може живитися від джерела 5.0 В за допомогою вбудованого LDO-регулятора, або безпосередньо від джерела 3.3 В. Внутрішній LDO не призначений для живлення інших зовнішніх мікросхем і потребує конденсатора ємністю 1 мкФ, підключеного до “землі”, як показано на Рис.3.12.

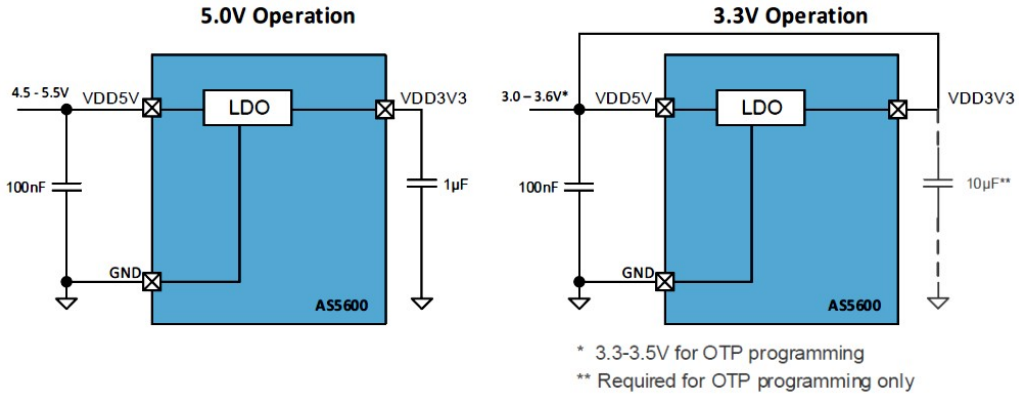


Рис. 3.13. Схема живлення сенсора AS5600

Для доступу до регістрів AS5600 використовуються дві адреси. Перша — це адреса підлеглого пристрою (slave address), яка використовується для вибору AS5600. Усі транзакції шини I²C включають адресу підлеглого пристрою. Адреса підлеглого пристрою AS5600 — 0x36 (0110110 у двійковому форматі). Друга адреса — це адреса слова (word address), що надсилається в першому байті під час транзакції запису.

Головний мікроконтролер (хост MCU) ініціює передачу даних. 7-бітна адреса підлеглого пристрою AS5600 становить 0x36 (0110110 у двійковому форматі).

Підтримувані режими. випадкове/послідовне читання, побайтове/сторінкове записування, автоматичне інкрементування (ANGLE), стандартний режим, швидкий режим, швидкий режим Plus.

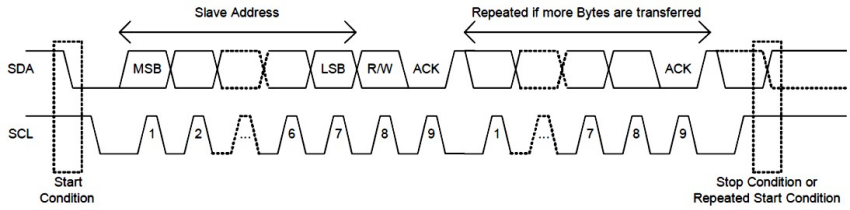
Сигнал SDA є двонаправленою лінією даних. Сигнал SCL — це тактовий сигнал, що генерується головним пристроєм шини I²C для синхронізації вибірки даних з лінії SDA. Максимальна частота SCL становить 1 МГц. Дані вибираються по висхідному фронту SCL. Для доступу до регістрів AS5600 використовуються дві адреси. Перша – це адреса підлеглого пристрою (slave address), яка використовується для вибору AS5600. Усі транзакції шини I²C містять адресу підлеглого пристрою. Адреса AS5600 – 0x36 (0110110 у двійковому форматі). Друга адреса – це адреса слова, що надсилається в першому байті при транзакції запису.

Адреса слова вибирає конкретний регістр на AS5600. Адреса слова завантажується в вказівник адреси на AS5600. Під час наступних транзакцій читання і наступних байтів у транзакції запису, вказівник адреси надає адресу вибраного регістра. Вказівник адреси інкрементується після кожного переданого байта, за винятком деяких транзакцій читання зі спеціальних регістрів.

Підтримуваний протокол шини. Передача даних може бути ініційована лише тоді, коли шина не зайнята. Під час передачі даних, лінія даних (SDA) повинна залишатися стабільною, коли SCL є високим. Зміни в стані лінії даних, коли SCL високий, інтерпретуються як умови СТАРТ або СТОП. Відповідно, були визначені наступні стани шини: шина не зайнята: обидві лінії SDA і SCL залишаються високими; старт передачі даних: зміна стану SDA з високого на низький, коли SCL є високим, визначає умову СТАРТ; стоп передачі даних: зміна стану SDA з низького на високий, коли SCL є високим, визначає умову СТОП. Кожна транзакція шини I²C починається з умови СТАРТ і завершується умовою СТОП. Кількість байтів даних, що передаються між умовами СТАРТ і СТОП, не обмежена та визначається головним

пристроєм шини I²C. Інформація передається побайтово, і кожен приймач підтверджує її дев’ятим бітом (Acknowledge bit).

Кожен підлеглий пристрій I²C, коли до нього звертаються, зобов'язаний генерувати підтвердження після прийому кожного байта. Головний пристрій шини I²C повинен генерувати додатковий тактовий імпульс для цього біта підтвердження. Підлеглий пристрій, що підтверджує прийом, повинен притягнути лінію SDA до “землі” під час тактового імпульсу підтвердження таким чином, щоб SDA був стабільним і низьким протягом високої фази тактового імпульсу підтвердження. Звісно, необхідно враховувати час встановлення та утримання. Головний пристрій повинен сигналізувати про закінчення транзакції читання, не генеруючи біт підтвердження на останньому байті, що був виведений з підлеглого пристрою. У цьому випадку підлеглий пристрій повинен залишити SDA високим, щоб дати можливість головному пристрою згенерувати умову СТОП.



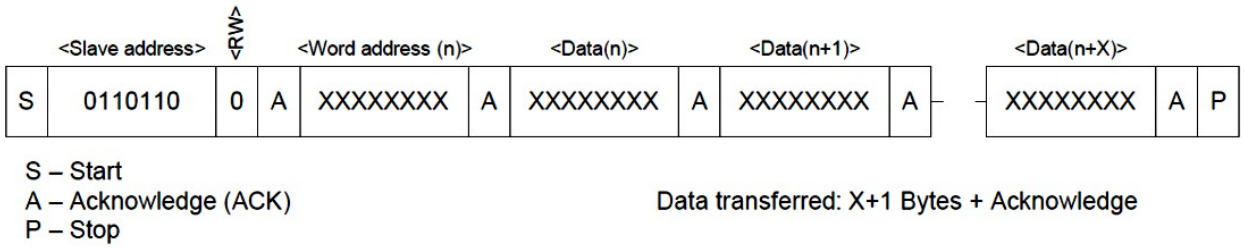
Залежно від стану біта R/W (Read/Write, читання/запис) можливі два типи передачі даних:

Передача даних від головного передавача до підлеглого приймача (запис). Перший байт, переданий головним пристроєм (master), — це адреса підлеглого пристрою (slave address), за якою йде R/W = 0 (запис). Далі йде певна кількість байтів даних. Підлеглий пристрій повертає біт підтвердження (ACK) після кожного отриманого байта. Якщо підлеглий пристрій не розуміє команду або дані, він надсилає біт відсутності підтвердження (NACK). Дані передаються, починаючи зі старшого біта (MSB).

Передача даних від підлеглого передавача до головного приймача (читання). Головний пристрій передає перший байт (адресу підлеглого пристрою). Підлеглий пристрій повертає біт підтвердження, після чого

підлеглий пристрій передає певну кількість байтів даних. Головний пристрій повертає біт підтвердження після всіх отриманих байтів, крім останнього. Наприкінці останнього отриманого байта повертається NACK. Головний пристрій генерує всі тактові імпульси SCL та умови СТАРТ і СТОП. Передача завершується умовою СТОП або повторною умовою СТАРТ. Оскільки повторна умова СТАРТ також є початком наступної послідовної передачі, шина не звільняється. Дані передаються, починаючи зі старшого біта (MSB).

Режим підлеглого приймача (режим запису). Послідовні дані та тактовий сигнал приймаються через SDA та SCL. За кожним байтом слідує біт підтвердження або відсутності підтвердження, залежно від того, чи вибирає вказівник адреси дійсну адресу. Умови СТАРТ і СТОП розпізнаються як початок і кінець транзакції шини. Байт адреси підлеглого пристрою — це перший байт, отриманий після умови СТАРТ. 7-бітна адреса AS5600 – 0x36 (0110110 у двійковому форматі). За 7-бітним адресом підлеглого пристрою йде біт напрямку (R/W), який для запису дорівнює 0 (низький рівень). Після отримання та декодування байта адреси підлеглого пристрою, підлеглий пристрій видає підтвердження на SDA. Після того як AS5600 підтверджує адресу підлеглого пристрою і біт запису, головний пристрій передає адресу регістра (word address) на AS5600. Вона завантажується в вказівник адреси на AS5600. Якщо адреса є дійсною, AS5600 відповідає, надсилаючи підтвердження (біт A низький). Якщо вказівник адреси вибирає недійсну адресу, надсилається відсутність підтвердження (біт A високий). Після цього головний пристрій може передати нуль або більше байтів даних. Якщо вказівник адреси вибирає недійсну адресу, отримані дані не зберігаються. Вказівник адреси буде інкрементуватися після передачі кожного байта, незалежно від того, дійсна адреса чи ні. Якщо вказівник адреси знову досягає дійсної позиції, AS5600 відповідає підтвердженням і зберігає дані. Головний пристрій генерує умову СТОП, щоб завершити транзакцію запису.



Режим підлеглого передавача (режим читання). Перший байт приймається та обробляється так само, як і в режимі підлеглого приймача. Однак у цьому режимі біт напрямку вказує, що AS5600 буде передавати дані на SDA. Умови СТАРТ і СТОП розпізнаються як початок і кінець транзакції на шині. Байт адреси підлеглого пристрою – це перший байт, отриманий після того, як головний пристрій згенерує умову СТАРТ. Байт адреси підлеглого пристрою містить 7-бітну адресу AS5600. За 7-бітною адресою підлеглого пристрою слідує біт напрямку (R/W), який для читання дорівнює 1 (високий рівень). Після отримання та декодування байта адреси підлеглого пристрою, підлеглий пристрій видає підтвердження на лінії SDA.

Після цього AS5600 починає передавати дані, починаючи з регістра, на який вказує вказівник адреси. Якщо вказівник адреси не був записаний перед ініціацією транзакції читання, першою адресою, яка буде прочитана, буде остання, збережена у вказівнику адреси. AS5600 має отримати біт відсутності підтвердження (NACK), щоб завершити транзакцію читання.

В таблиці 3.7. наведено інформацію про регістри AS5600.

Таблиця 3.6. Регістри AS5600

Address	Name	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Configuration Registers										
0x00	ZMCO	R							ZMCO(1:0)	
0x01	ZPOS	R/W/P					ZPOS(11:8)			
0x02			ZPOS(7:0)							
0x03	MPOS	R/W/P					MPOS(11:8)			
0x04			MPOS(7:0)							
0x05	MANG	R/W/P					MANG(11:8)			
0x06			MANG(7:0)							
0x07	CONF	R/W/P			WD	FTH(2:0)			SF(1:0)	
0x08			PWMF(1:0)		OUTS(1:0)		HYST(1:0)		PM(1:0)	

Output Registers											
0x0C	RAW ANGLE	R					RAW ANGLE(11:8)				
0x0D		R	RAW ANGLE(7:0)								
0x0E	ANGLE	R					ANGLE(11:8)				
0x0F		R	ANGLE(7:0)								
Status Registers											
0x0B	STATUS	R			MD	ML	MH				
0x1A	AGC	R	AGC(7:0)								
0x1B	MAGNITUDE	R					MAGNITUDE (11:8)				
0x1C		R	MAGNITUDE(7:0)								
Burn Commands											
0xFF	BURN	W	Burn_Angle = 0x80; Burn_Setting = 0x40								

Регістри ZPOS/MPOS/MANG. Ці регістри використовуються для налаштування початкової позиції (ZPOS) та кінцевої позиції (MPOS) або максимального кута (MANG) для створення вужчого кутового діапазону. За замовчуванням повний діапазон становить від 0 до 360 градусів.

Регістр CONF використовується для налаштування AS5600. Регістри ANGLE/RAW ANGLE – містить немасштабований і незмінений кут. Масштабоване та відфільтроване вихідне значення доступне в регістрі ANGLE. Регістр STATUS – містить біти, які показують поточний стан AS5600. Регістр MAGNITUDE – показує значення амплітуди внутрішнього блоку CORDIC.

Аналоговий вихідний режим. За замовчуванням, вихідний каскад AS5600 налаштований на аналоговий співвідносний вихід. Цифро-аналоговий перетворювач (ЦАП) має 12-бітну роздільну здатність. Для спрощення процесу розробки доцільно використовувати готовий апаратний модуль з магнітним сенсором AS5600 (Рис. 3.14).

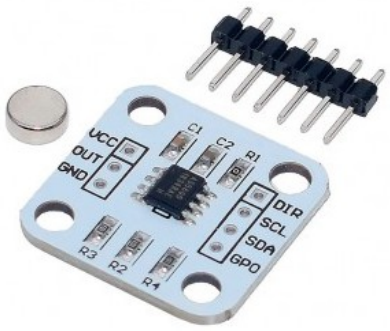


Рис. 3.14. Модуль з магнітним сенсором AS5600

Схему підключення модуля до плати ARDUINO UNO наведено на Рис.3.15.

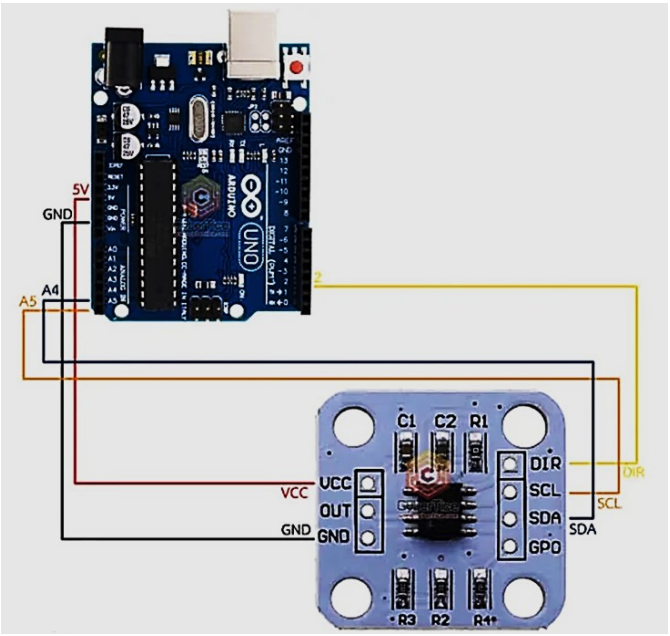


Рис. 3.15. Схема підключення модуля з AS5600 до плати ARDUINO UNO

Опис алфавітно-цифрового РКД на базі контролера HD44780. Рідкокристалічні дисплеї (РКД) належать до категорії доступних компонентів для візуалізації текстових та числових даних. Наявність інтегрованого модуля підсвічування забезпечує чітке зчитування символів навіть в умовах низької освітленості. Завдяки широкому температурному діапазону експлуатації (від −20 °C до +70 °C), дані індикатори є надійним рішенням для використання у складі мобільної та польової електроніки. Конструктивно типовий РК-модуль об’єднує в собі два ключові елементи:

Керуючий контролер (найчастіше сумісний із галузевим стандартом HD44780) та рідкокристалічну панель, що безпосередньо формує зображення. Для забезпечення видимості в умовах обмеженого освітлення пристрій обладнано системою світлодіодного (LED) підсвічування. На Рис.3.5 продемонстровано зовнішній вигляд модуля рідкокристалічного дисплея. Дисплей дозволяє одночасно відображати до 32 символів, розподілених у два рядки по 16 позицій у кожному. Кожне знакомісце базується на піксельній матриці розміром 5×8 точок. Для кращого сприйняття тексту між сусідніми символами передбачено технологічний проміжок шириною в один піксель.

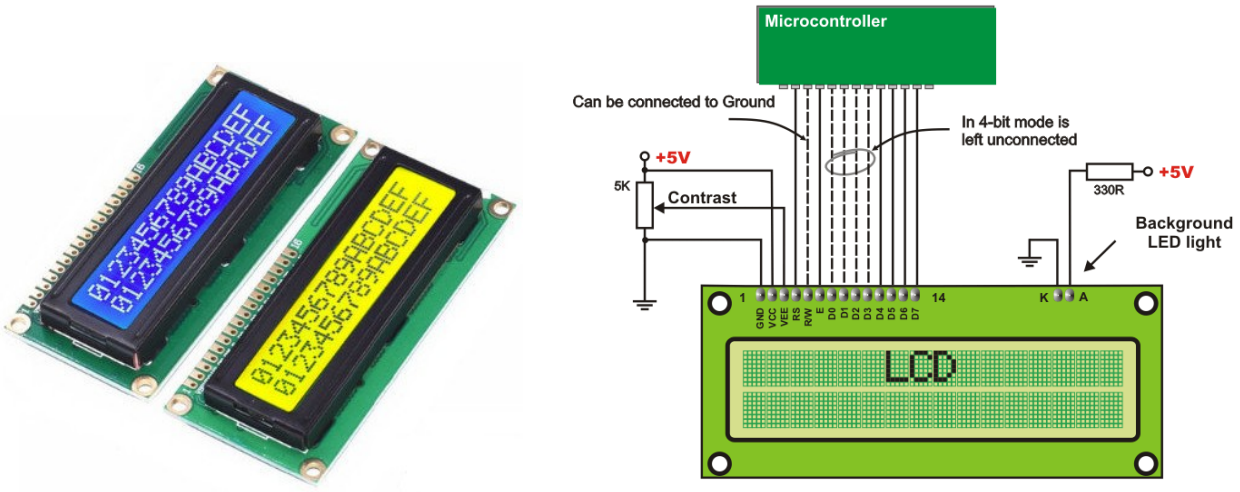


Рис. 3.16. Алфавітно-цифровий РКД 16×2 з контролером HD44780 та схема його підключення

Візуалізація кожного знака на екрані здійснюється шляхом зіставлення його унікального коду, що зберігається в осередку оперативної пам'яті (DDRAM) модуля, з відповідним графічним образом. Архітектура пристрою включає логіку керування панеллю та два типи пам’яті: знакогенератор (ПЗП для стандартних символів) та пам'ять користувача (для створення власних символів).

Ключові технічні характеристики модуля. Мовна підтримка: Інтегрований знакогенератор містить дві сторінки кодових таблиць, що дозволяють відображати символи латиниці та кирилиці. Модуль підтримує два режими передачі даних — через 4-бітну або 8-бітну шину. Пристрій здатний

інтерпретувати команди, що надходять через шину даних, а також виконувати операції запису та зчитування інформації з ОЗП.

Передбачено можливість зберігання та виведення до 8 унікальних графічних образів, розроблених користувачем самостійно. Підтримка декількох режимів роботи курсора, включаючи статичне відображення та режим миготіння. Конструкція модуля дозволяє програмно або апаратно регулювати інтенсивність підсвічування та рівень контрастності зображення.

Принципи керування та архітектура контролера HD44780. Функціонування РК-модуля базується на логіці контролера Hitachi HD44780. Взаємодія з ним здійснюється через систему регістрів, де ключову роль відіграють регістр інструкцій (IR) та регістр даних (DR).

Механізм адресації. Керування вибором регістрів реалізовано через лінію RS (*Register Select*): при низькому рівні сигналу (RS=0) активується регістр команд (IR), що дозволяє надсилати інструкції для дешифрації та виконання; при високому рівні (RS=1) доступ до регістра даних (DR).

Залежно від конфігурації, через DR інформація записується у відеопам’ять (DDRAM) або в область пам'яті знакогенератора (CGRAM). Точне місце запису визначається внутрішнім лічильником адреси (AC).

Організація пам’яті та драйверна логіка. Внутрішня відеопам’ять (DDRAM) об'ємом 80 байт призначена для зберігання ASCII-кодів символів. Архітектурно вона розділена на два рядки по 40 сегментів. Важливо зауважити, що незалежно від фізичної геометрії самого дисплея (наприклад, 16x2 або 20x4), логічна адресація завжди сприймає пам’ять як два рядки по 40 символів. Контролер працює за принципом динамічної індикації, безперервно оновлюючи стан рідкокристалічної матриці. Власних ресурсів чипа HD44780 достатньо для керування 40 сегментами (SEG1–SEG40), що дозволяє обслуговувати панелі форматом до 8 символів. Для дисплеїв з більшою місткістю використовуються допоміжні драйвери (наприклад, HD44100), кожен з яких додає підтримку ще 40 сегментів.

Для розробки програмного забезпечення під платформу Arduino застосовується стандартна бібліотека LiquidCrystal, прототипи функцій якої описані в однойменному заголовному файлі LiquidCrystal.h.

На Рис.3.5 представлена електрична схема з'єднання РК-модуля з мікроконтролерною платою Arduino Uno.

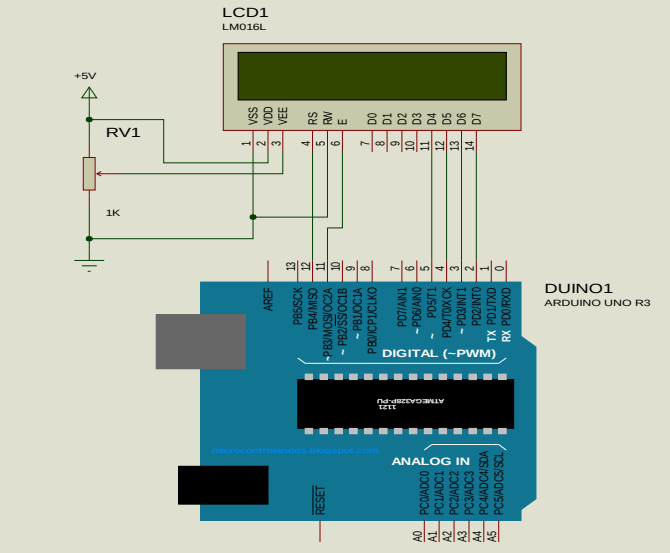


Рис. 3.17. Схема підключення РКД до Arduino UNO

Огляд модулів з (RTC) DS1307. Для забезпечення точного відліку часу в системі трекера доцільно використовувати мікросхему DS1307 або готові рішення на її базі. Ця мікросхема є спеціалізованим годинником реального часу (RTC), що підтримує обмін даними через послідовну шину I2C. В межах даної розробки було обрано комбінований модуль DS1307, який додатково оснащений мікросхемою енергонезалежної пам'яті AT24C32 (EEPROM). Таке поєднання дозволяє не лише відстежувати часові параметри, а й зберігати критично важливі налаштування пристрою. Обраний модуль (зображений нижче) є оптимальним за співвідношенням функціональності та вартості.

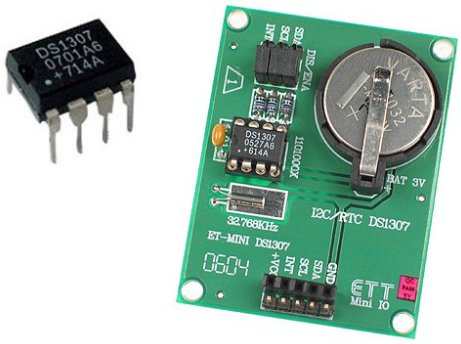


Рис. 3.18. Мікросхема та модуль годинника реального часу DS1307

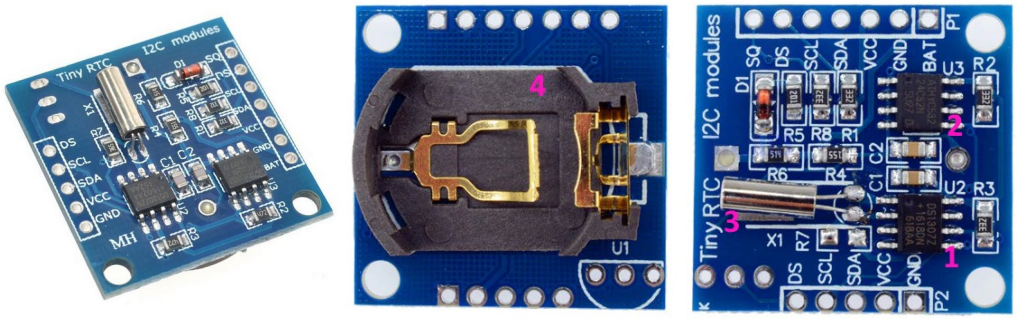
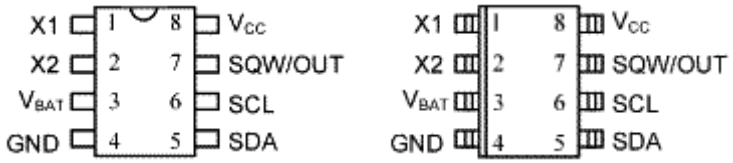


Рис. 3.19. Модуль DS1307 з AT24C32 EEPROM

Цей модуль базується на поєднанні двох IC, що взаємодіють через шину I2C: годинника реального часу DS1307 та енергонезалежної пам'яті AT24C32 об'ємом 32 Кбіт (4 КБ). Для забезпечення автономної роботи системи (збереження ходу часу при відключенні зовнішнього живлення) передбачено тримач для елемента живлення типу CR2032.

Характеристики модуля з DS1307. Пристрій автоматично реєструє секунди, хвилини, години, а також дату, місяць, день тижня та рік. Вбудований алгоритм автоматично компенсує високосні роки аж до 2100 року. Передбачено програмований вивід SQW, здатний генерувати прямокутні імпульси для синхронізації роботи зовнішніх периферійних пристроїв. При живленні від резервного джерела (батареї) споживання струму становить менше 500 нА, при цьому внутрішній тактовий генератор продовжує функціонувати. Мікросхема стабільно працює в широкому температурному діапазоні: від -40 ° С до +85 °С. Мікросхема доступна у стандартних 8-вивідних корпусах типу DIP або SOIC.



а) б)

Рис. 3.20. Розташування виводів DS1307: а) корпус DIP б) корпус SOIC

Призначення виводів DS1307. VCC: Вхід основного живлення інтегральної схеми. X1, X2: Контакти для підключення зовнішнього резонатора з частотою 32,768 кГц. VBAT: Вхід для підключення резервного джерела живлення напругою +3 В. GND: Загальний провід (земля). SDA (Serial Data): Двонапрямлена лінія передачі даних послідовного інтерфейсу. SCL (Serial Clock): Лінія тактування (синхронізації) послідовного інтерфейсу. SQW/OUT (Square Wave/Output Driver): Вихідний каскад, що генерує сигнал прямокутної форми.

Для функціонування виводу SQW/OUT необхідне використання зовнішнього підтягуючого резистора номіналом 4,7 кОм, підключеного до лінії +5 В. Цей вихід використовується для генерації сигналів точної частоти: 1 Гц, 4 кГц, 8 кГц або 32 кГц.

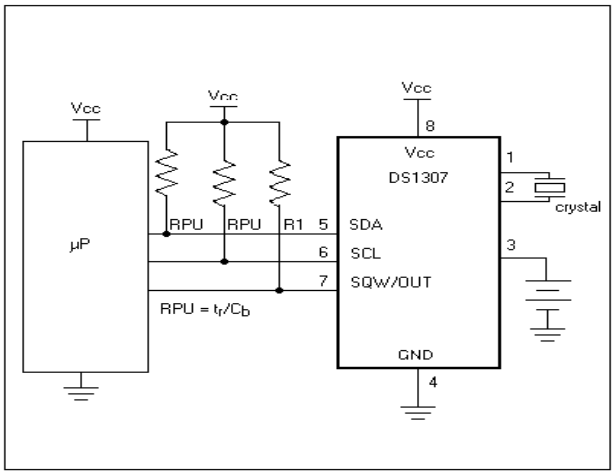


Рис. 3.21 Типова схема під'єднання DS1307

Інформацію від годинника отримуємо зчитуванням відповідних байтів регістрів. Регістри DS1307 наведено в Табл.3.7.

Таблиця 3.7. Регістри DS1307

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Налаштування часу та календарних даних здійснюється шляхом запису відповідних значень у внутрішні регістри мікросхеми. Важливо зауважити, що всі дані зберігаються та обробляються у двійково-десятковому форматі (BCD). За зупинку або запуск годинника відповідає 7-й біт регістра з адресою 0x00, відомий як CH (Clock Halt): CH=1: Тактовий генератор зупинено (режим зниженого споживання). CH=0: Тактовий генератор активовано.

Після подачі живлення стан регістрів є невизначеним. Тому процедура ініціалізації обов’язково повинна включати примусовий запуск генератора шляхом скидання біта CH у “0”.

Формати представлення часу. DS1307 підтримує роботу в двох режимах, вибір яких здійснюється 6-м бітом регістра годин: 24-годинний формат (Біт6=0): Біт5 використовується для відображення десятків годин (діапазон 20–23). 12-годинний формат (Біт6=1): Біт 5 функціонує як індикатор AM/PM (високий рівень відповідає PM).

За допомогою регістра керування здійснюється налаштування режиму роботи виводу SQW/OUT, зокрема, вибір частоти вихідного сигналу або переведення його у режим загального виходу.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

Bit 7: Output Control (OUT) – встановлює логічний рівень на виході SQW pin.7 при Bit 4= 0 і Bit 7=1 то на виході SQW=лог.1. (якщо Bit 7=0 на виході SQW=лог.0). Bit 4: Square-Wave Enable (SQWE) – Bit 4=1, то на виході SQW

імпульси частотою (1 Hz, 4 KHz, 8 KHz, 32 KHz) (RS0 і RS1 bit). (SQWE=1, RS0=0 і RS1=0 то частота=1 Hz). Управління і налаштування RTC DS1307 здійснюється по інтерфейсу I2C. DS1307 на послідовній шині працює як ведений пристрій.

3.2. Проєктування трекера сонячної електростанції для автономних блокпостів поліції в САПР Proteus VSM

Розроблений трекер сонячної електростанції для блокпостів поліції призначений для автономного позиціонування сонячної панелі мобільної електростанції перпендикулярно до Сонця протягом дня. Обчислення азимуту та альтитуди Сонця для вказаної довготи і широти дати і години, та вмикання виконавчих моторів для орієнтації панелі. Обчислення часу сходу та заходу Сонця та відповідних азимутів. Результати обчислень виводяться на рідкокристалічний дисплей. Клавітура дозволяє здійснити попередні налаштування: довготи, широти, дати, години.

На Рис.3.22 представлена запропонована структура трекера. Система позиціонування об’єднує як апаратне, так і програмне забезпечення. Апаратне забезпечення трекера включає в себе: Апаратну платформу Arduino UNO з МК AVR (ATmega328), який виконує основні функції обробки даних та керування; сенсор азимуту AS5600, давач альтитуди троьох-осьовий акселерометр ADXL345, вони утворюють систему збору даних, які передаються на обробку до МК; годинник реального часу DS1307 контролює дату і час, LCD служить для відображення інформації.

Програмне забезпечення відповідає за зчитування даних з датчиків, їхню обробку відповідно до заданих алгоритмів (наприклад, порівняння обчислених азимуту і альтитуди з реальними, прийняття рішень щодо керування виконавчими пристроями), та виведення інформації на LCD та в послідовний порт.

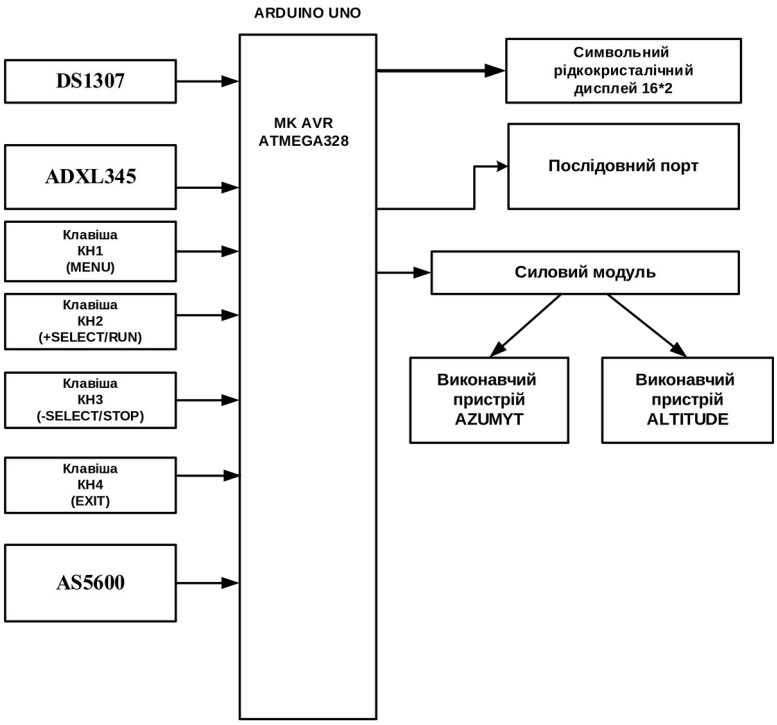


Рис. 3.22. Загальна структура трекера сонячної електростанції

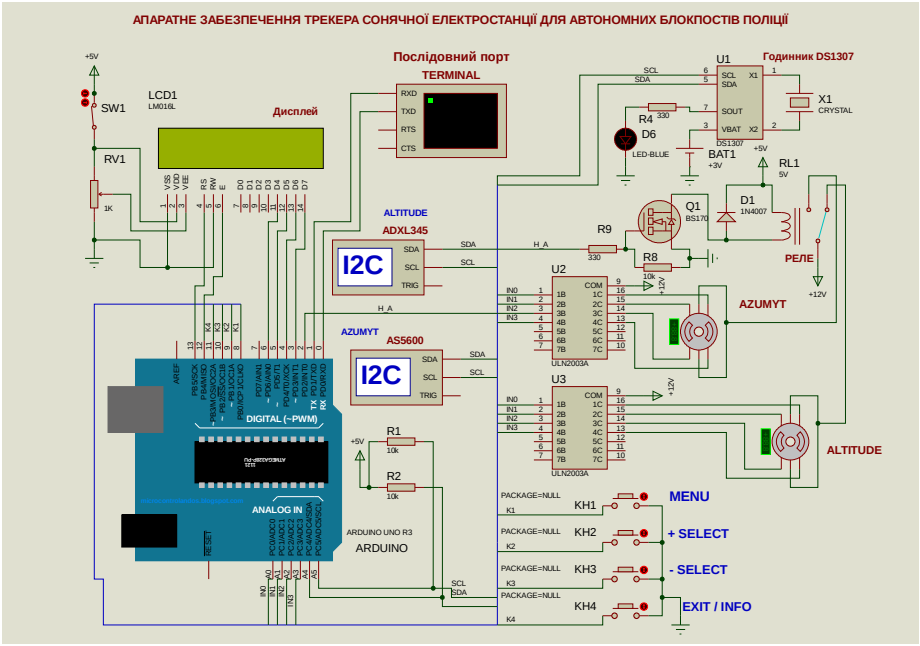


Рис. 3.23. Апаратне забезпечення трекера сонячної електростанції спроектоване в САПР Proteus VSM

Мікроконтролер здійснює обробку отриманих значень від сенсорів обчислює де знаходиться Сонце, після чого виводить цю інформацію на LCD і

дає команди виконавчим пристроям для позиціонування сонячної панелі електростанції. На Рис.3.23 представлена схема спроектованої апаратної частини трекера сонячної електростанції для автономних блок постів поліції, розроблена в середовищі автоматизованого проектування та моделювання електронних схем та програмованих пристроїв Proteus VSM. Згідно зі схемою підключення, всі цифрові давачі, що використовують інтерфейс I2C, під'єднані до портів A4, A5 плати ARDUINO UNO. LCD підключено до виводів порту D МК. Конкретне призначення виводів порту A (наприклад, для ліній даних, ліній керування RS, EN, RW) є стандартним для підключення LCD-модулів на базі контролера HD44780. Клавіші налаштування параметрів трекера підключені наступним чином: клавіша MENU/ENTER для входу в головне меню та підтвердження вибору підключена до виводу PB0 порту B; клавіша SELECT+ для вибору або збільшення значення підключена до виводу PB1 порту B; клавіша SELECT- для вибору або зменшення значення підключена до виводу PB2 порту B; клавіша EXIT виходу з меню в робочий режим підключена до виводу PB3 порту B. Виконавчі пристрої (крокові двигуни) підєднано через силові мікросхеми U1 і U2. Послідовний інтерфейс використовує піни PD0, PD1. Така конфігурація дозволяє МК отримувати дані про позиціонування сонячної панелі з двох датчиків (AS5600, ADXL345), відображати інформацію на LCD-дисплеї, взаємодіяти з користувачем через набір кнопок для налаштування параметрів роботи трекера та передавати інформацію в послідовний порт. Сигнали управління виконавчими пристроями поступають на силові мікросхеми U1, U2 та ключ Q1 для включення реле, котрі в свою чергу вмикають відповідні крокові двигуни. Така практика використання реле забезпечує гальванічну розв'язку і запобігає попадання високої напруги на плату управління з МК.

Алгоритм роботи трекера дозволяє налаштовувати час, дату, координати розташування сонячної електростанції, часові пояси з допомогою клавіш МЕНЮ.

РОЗДІЛ 4
ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ТРЕКЕРА СОНЯЧНОЇ ЕЛЕКТРОСТАНЦІЇ ДЛЯ АВТОНОМНИХ БЛОКПОСТІВ ПОЛІЦІЇ

4.1. Алгоритм роботи трекера сонячної електростанції для автономних блокпостів поліції

На Рис.4.1-4.7 зображено блок-схему алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції.

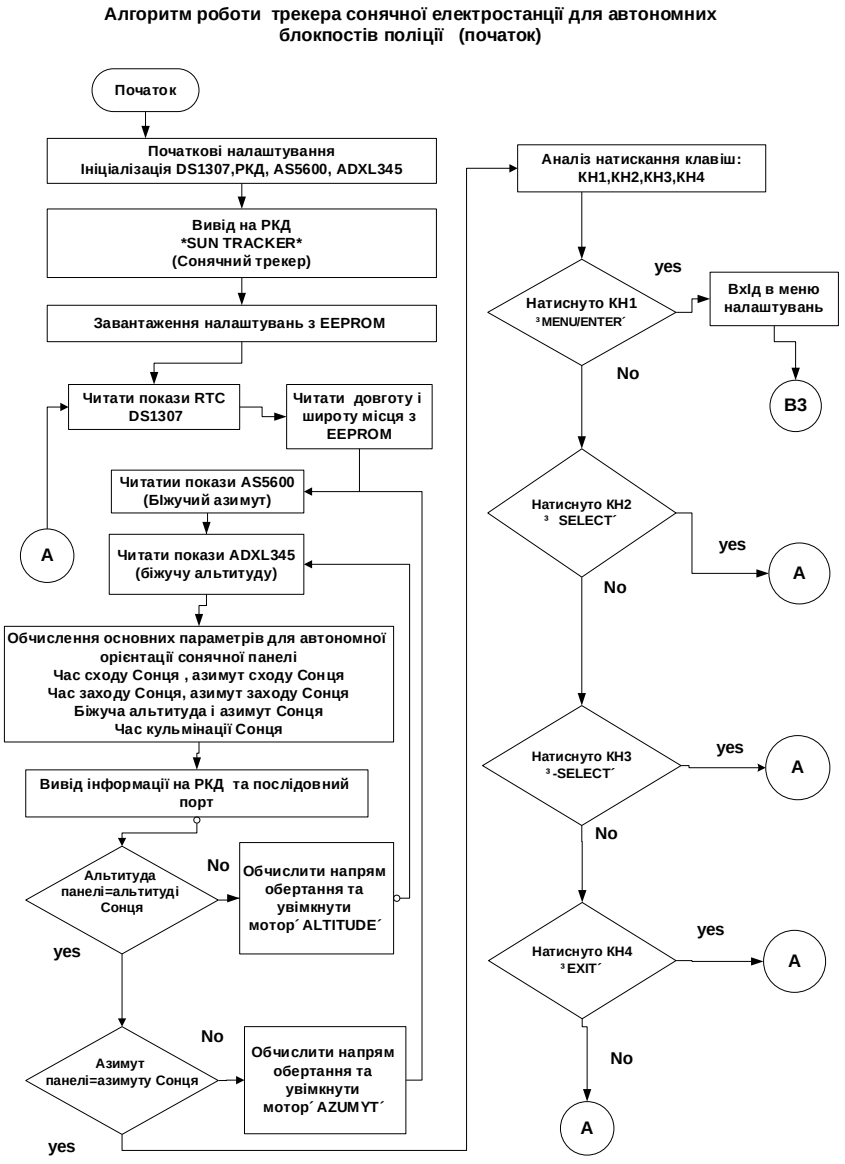


Рис. 4.1. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (початок)

Алгоритм роботи трекера сонячної електростанції для автономних блокпостів поліції. (продовження 1)

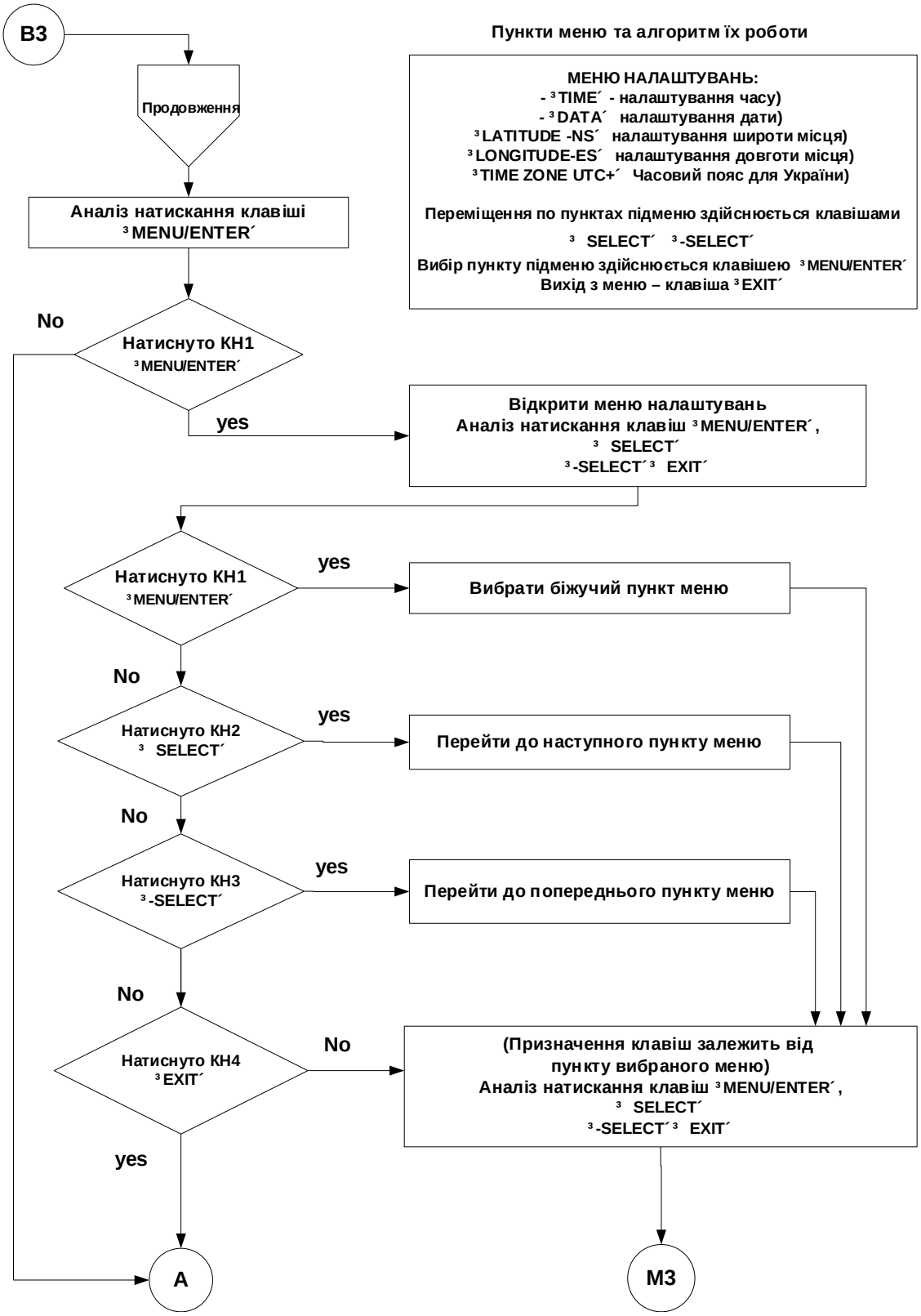


Рис. 4.2. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (продовження 1)

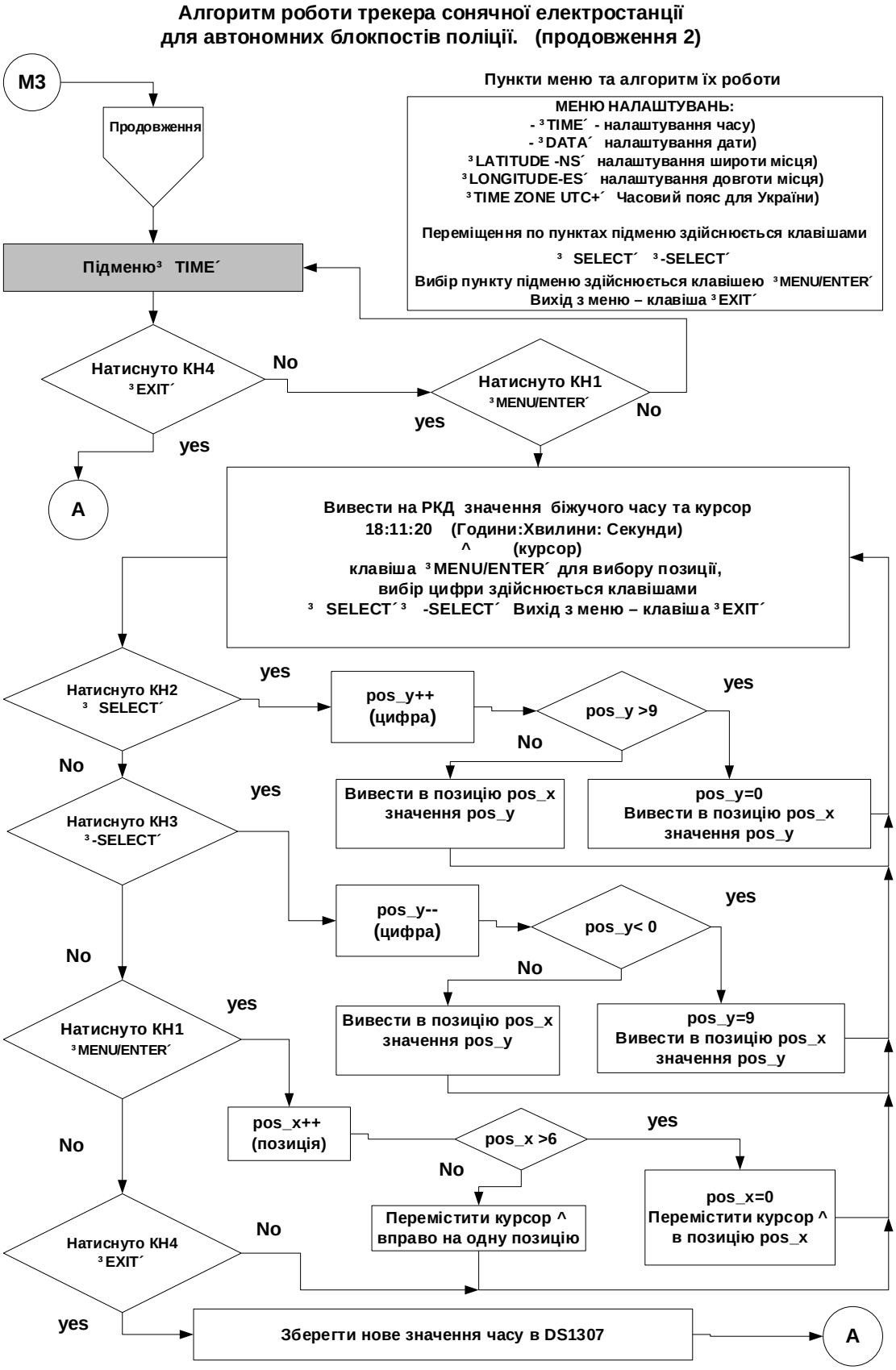


Рис. 4.3. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (продовження 2)

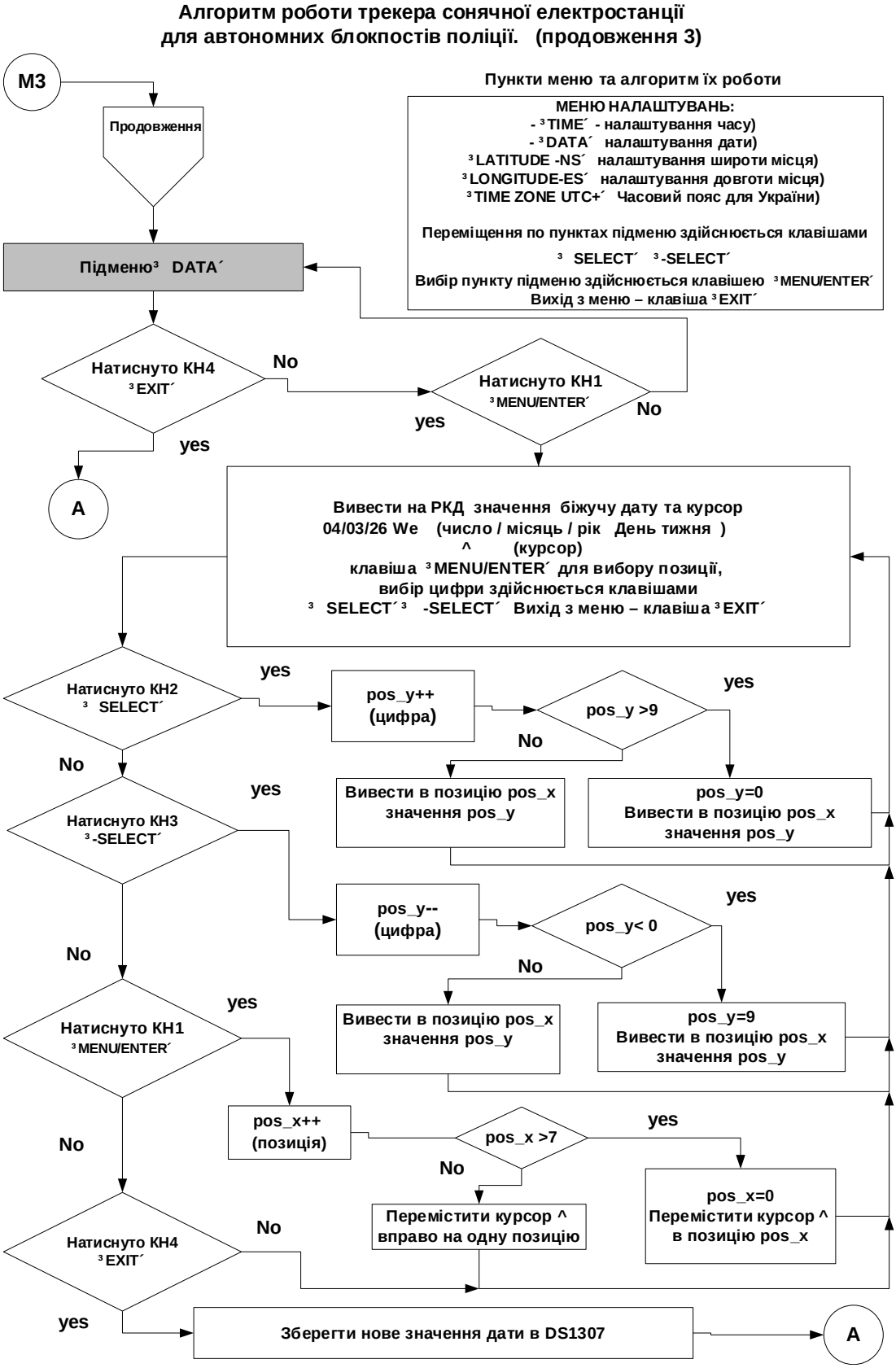


Рис. 4.4. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (продовження 3)

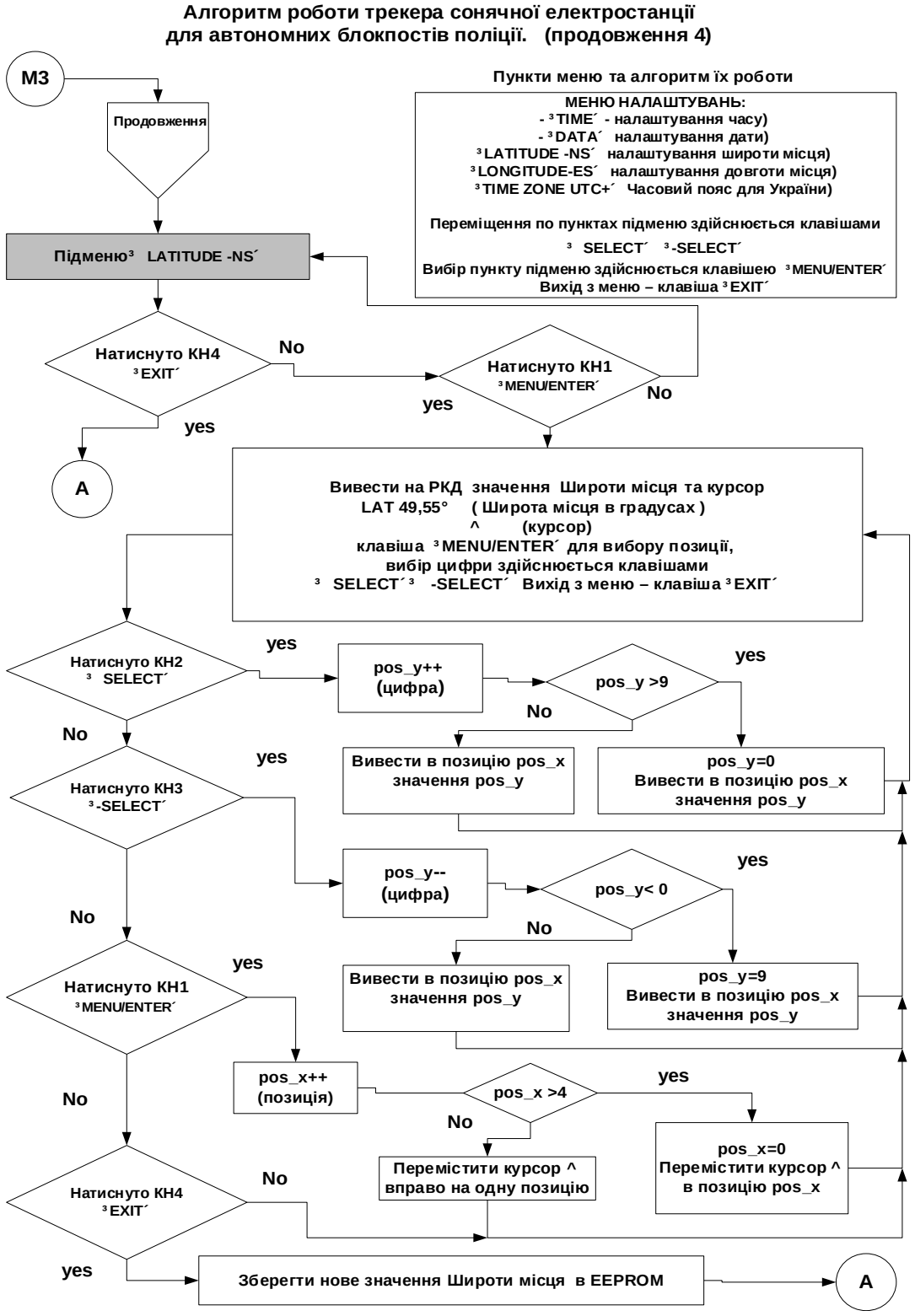


Рис. 4.5. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (продовження 4)

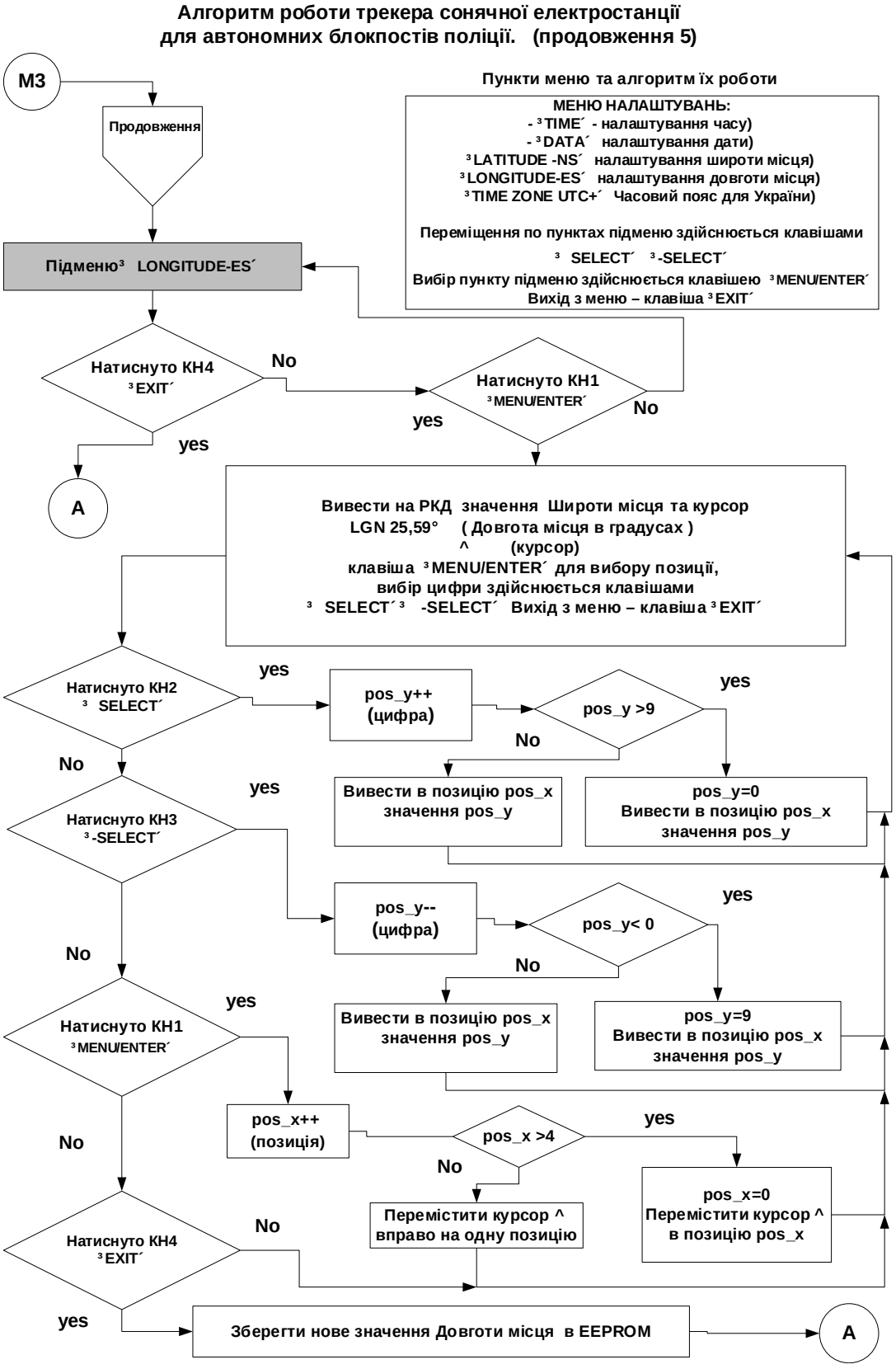


Рис. 4.6. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (продовження 5)

Алгоритм роботи трекера сонячної електростанції для автономних блокпостів поліції. (кінець)

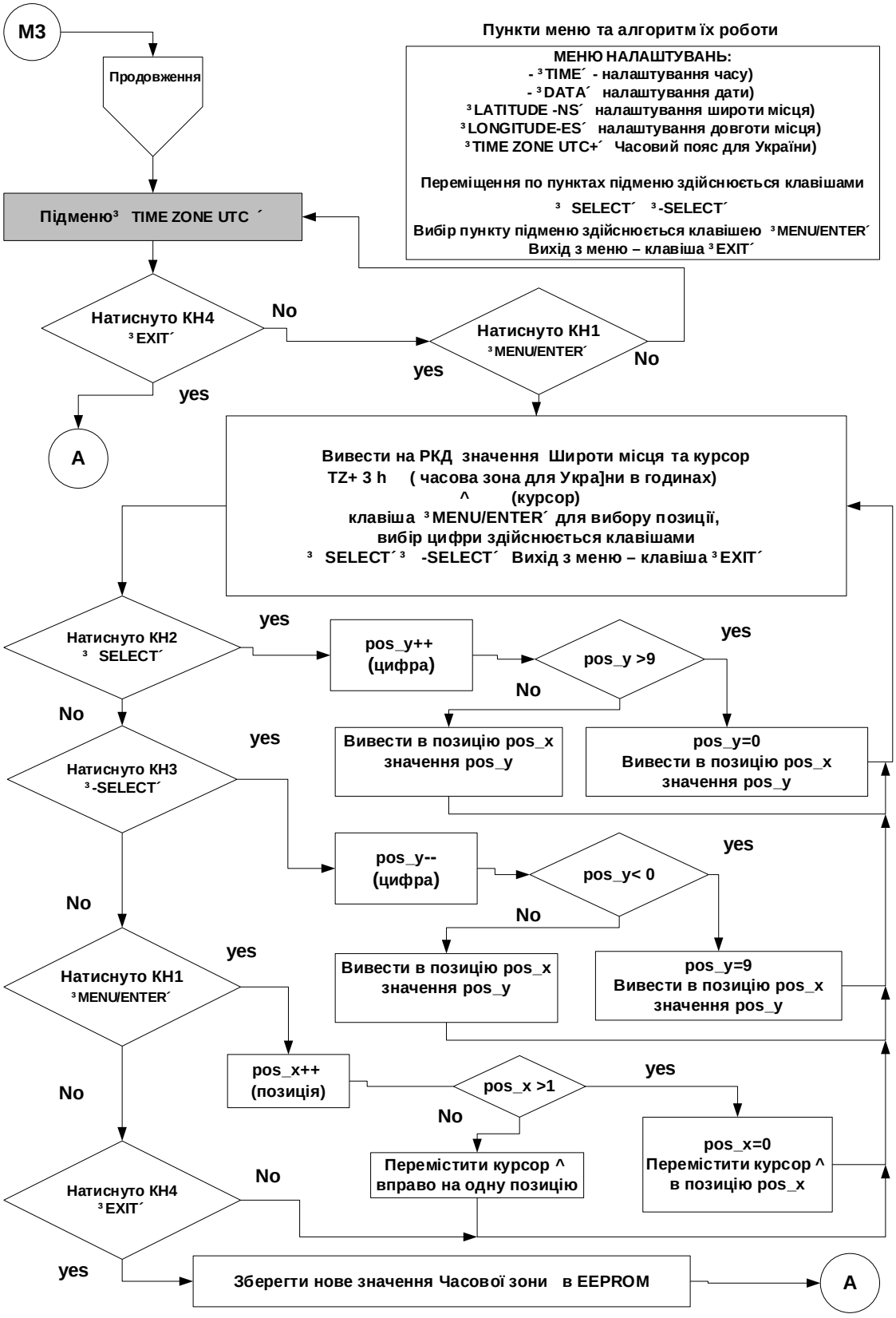


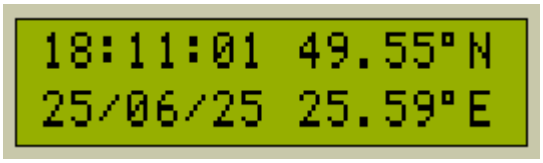
Рис. 4.7. Алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції (кінець)

Після подачі живлення мікроконтролер AVR розпочинає виконання вбудованого програмного забезпечення . На початковому етапі відбувається ініціалізація внутрішніх програмних змінних. Далі запускається процес ініціалізації сенсорів ADXL345, AS5600 годинника реального часу DS1307, ініціалізація рідкокристалічного дисплея (РКД) у розроблюваній системі здійснюється викликом функції lcd.begin(16, 2). Після завершення процесу ініціалізації, система виводить текстове повідомлення на екран РКД:

“SUN TRACKER” Далі розпочинається обчислення за відомими довготою, широтою місця та датою і часом азимуту і альтитуди Сонця. Інформація виводиться на символьний LCD екран та в послідовний порт.

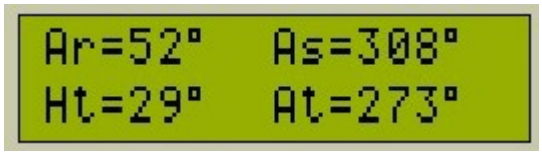
На РКД послідовно виводиться наступна інформація:

- година, дата, широта, довгота місця:



- час сходження Сонця ↑, час заходу Сонця ↓, * кульмінація Сонця, дата:

- азимут сходження Сонця (Ar), азимут заходу Сонця (As),біжуча альтитуда(Ht), біжучий азимут (At)



В послідовний порт передаються аналогічна інформація (Рис. 4.8): LOCATION – координати місця (широта, довгота); година, день, місяць, рік; EEST –час в Україні; UTC – час по Грінвічу; AST – місцевий зоряний час; Sun culmination – час кульмінації Сонця для даного місця; Sunrise – час сходження Сонця на вказану дату; Sunset - час заходу Сонця на вказану дату; Az Sunrise – азимут сходження Сонця на вказану дату; Az Sunset - азимут заходу Сонця на

вказану дату; Azt – митєвий азимут (використовується для позиціонування); Altitude – митєва альтитуда (використовується для позиціонування).

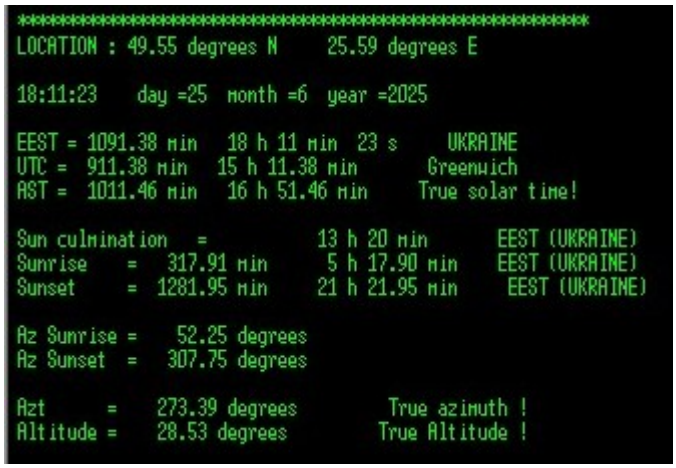


Рис. 4.8. Інформація з послідовного порта

МК отримує інформацію з давачів азимуту (AS5600) і альтитуди (ADXL345), та приймає рішення на вмикання моторів для позиціонування сонячної панелі згідно обчислених Azt і Altitude. МК періодично опитує клавіші меню і при їх натисканні викликає відповідні функції обробки.

4.2. Розробка програмного модуля для роботи з цифровим акселерометром ADXL345, енкодером AS5600, годинником реального часу DS1307

Для забезпечення взаємодії з цифровим сенсором ADXL345 було створено наступні функції:

void ADXL345_Init (void) – функція ініціалізації ADXL345
void readAccel() – функція читання даних з регістрів ADXL345
Функції обчислення кутів нахилу сонячної панелі по осях XOY
// Roll (Крен) atan2(y, x) повертає кут у радіанах
roll_rad=atan2(ya, za); roll_deg=roll_rad*180.0/PI; // Перетворюємо радіани в градуси
// кут Pitch (Тангаж)
pitch_rad=atan2(-xa, sqrt(ya*ya+za*za));
pitch_deg=pitch_rad*180.0/PI; // Перетворюємо радіани в градуси

Для роботи з годинником реального часу DS1307 створено функції :

// Функція налаштування годинника на дату
void SetDate(byte d, byte m, byte y, byte w);
// Функція налаштування годинника на час
SetTime(d_hour, d_minute, d_second);
// Функція повертає годину, хвилини і секунди у форматі BCD
void DS1307_GetTime(byte *hours, byte *minutes, byte *seconds)
// Функція повертає місяці, дні і рік у форматі BCD.

void DS1307_GetDate(byte *months, byte *days, byte *years, byte *dayofwk)
void readDATE() // функція читає дату
void readTIME() // функція читає час
Для роботи з магнітним енкодером створено функції:
// функція читати статус магнітометра as5600
int16_t as5600_read_status(void)
// функція читати значення сили магнітного поля
int16_t as5600_read_magnitude(void) //12-біт значення магнітного поля(0-4095)
// функція читати значення чутливості до магнітного поля
int16_t as5600_read_agc(void) // Значення AGC (0-255), або -1 у випадку помилки
// функція читати сире значення кута з AS5600
int16_t as5600_read_raw_angle(void) // Зчитує сире 12-бітне значення кута з AS5600
// функція читає відфільтроване значення кута з AS5600
int16_t as5600_read_angle(void)

Лістинг програмного коду для роботи з цифровими давачами ADXL345, DS1307, AS5600 наведено у додатку 1.

4.3. Розробка програмного модуля головного меню для автономної орієнтації сонячної панелі електростанції за Сонцем

Для складних обчислень митєвого азимуту і альтитуди Сонця створено наступні функції:

Функція latitude() – широта Північна (від +0....89,99 градусів)
void latns(void) – підменю налаштування широти спостереження
// Функція longitude – довгота Східна (від +0....89,99 градусів)
void lngew(void) – підменю налаштування довготи спостереження
Функція timezone – часова зона (від +0....6 годин до Монголії)
void timezone (void) – підменю налаштування часової зони

В тілі головного програмного модуля проводяться обчислення: рівняння часу EOT, істинного Сонячного Часу (AST) з врахуванням всіх поправок, обчислення схилення Сонця, обчислення годинного кута, обчислення висоти Сонця над горизонтом Altitude, обчислення годинного кута сходу Сонця, обчислення тривалості дня і часу сходу-заходу Сонця, час кульмінації Сонця на вказану дату, час сходу Сонця, час заходу Сонця, азимут сходу-заходу Сонця, митєвий азимут Сонця.

Якщо натиснуто кнопку MENU_ENTER відбувається очистка екрану РКД і вивід головного меню. З допомогою клавіш меню (menu/enter, select+, select-, exit) можна налаштувати час, дату, довготу, широту місця часовий пояс. Відкореговані значення зберігаються в енергонезалежній пам’яті і при

вимиканні живлення системи – зберігаються. Клавішами select+, select- редагуємо вибраний параметр і натискаємо клавішу menu/enter. Для виходу з меню налаштувань у робочий режим тиснемо клавішу exit.

4.4. Програмна реалізація головного алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції

Програма реалізує алгоритм функціонування трекера сонячної електростанції для автономних блокпостів поліції згідно Рис.4.1. Програмний код головного алгоритму роботи наведено у додатку 1.

4.5. Моделювання та аналіз результатів роботи трекера сонячної електростанції для автономних блокпостів поліції в Proteus ISIS

Результатом компіляції коду для мікроконтролерів сімейства AVR є HEX-файл. Саме цей формат даних є необхідним для безпосереднього програмування мікросхеми. Для верифікації розробленого пристрою та тестування логіки роботи програмного забезпечення найдоцільніше використовувати середовище Proteus VSM. Це комплексне рішення від компанії Labcenter Electronics дозволяє проводити глибоке налагодження навіть складних систем, що включають мережу мікроконтролерів, процесорів або цифрових сигнальних обробників (DSP). Архітектура пакета Proteus розділена на два основні інструменти:

- ISIS: робоча зона для креслення принципових схем та їх віртуального тестування в режимі реального часу.
- ARES: спеціалізований модуль для проектування друкованих плат (PCB), що дозволяє перейти від симуляції до фізичного виконання проекту.

Детальні результати моделювання роботи спроєктованого цифрового пристрою наведено в додатку 2.

ВИСНОВКИ

У кваліфікаційній роботі представлено розробку апаратного та програмного забезпечення трекера сонячної електростанції для автономних блокпостів поліції. Створена система здійснює автономне позиціонування сонячної панелі електростанції перпедикулярно до Сонця в реальному масштабі часу. Однією з ключових особливостей розробленої системи є наявність вбудованого меню, яке надає користувачеві можливість конфігурувати порогові значення.

Трекер функціонує за таким алгоритмом, що підтримує стабільне позиціонування сонячної панелі в установленому діапазоні, запобігаючи її виходу за допустимі межі. Усі параметри конфігурації трекера сонячної електростанції для автономних блокпостів поліції зберігаються в енергонезалежній пам’яті МК, що дозволяє системі після перезапуску відновлювати попередні налаштування без необхідності їх повторного введення. Завдяки моніторингу параметрів через послідовний порт можна швидко реагувати на позаштатні ситуації, і досягати високої надійності системи позиціонування..

У процесі створення цифрового трекера сонячної електростанції для автономних блокпостів поліції застосовано актуальну електронну компонентну базу. Ключовими складниками системи є плата ARDUINO UNO, цифрові датчики ADXL345, AS5600, годинник реального часу DS1307 символьний дисплей, силовий виконавчий блок з потужними ключами, крокові двигуни.

У середовищі Proteus VSM розроблено електричну принципову схему а також створено модель цифрового трекера сонячної електростанції. Її алгоритм роботи та програмне забезпечення для Arduino UNO було розроблено мовою С в середовищі Arduino IDE.

Створено програмні модулі для взаємодії з цифровими сенсорами ADXL345, AS5600. Проведено імітаційне моделювання функціонування

системи в середовищі Proteus ISIS. Отримані результати моделювання підтвердили правильність програмної реалізації основного алгоритму роботи трекера сонячної електростанції для автономних блокпостів поліції та розроблених програмних модулів.

Під час виконання кваліфікаційного проєкту було поглиблено теоретичну базу та вдосконалено практичні навики у сфері проектування мікроконтролерних систем. Зокрема, значну увагу було приділено розробці ефективних алгоритмів, написанню прикладного програмного забезпечення для вбудованих рішень, а також методам його тестування та налагодження.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. САПР Proteus VSM: Офіційна сторінка завантаження інструментів моделювання Labcenter Electronics. [Електронний ресурс]. — Режим доступу: https://www.labcenter.com/downloads/.
2. AS5600L Datasheet: Технічна специфікація [Електронний ресурс]. – Режим доступу: Microchip Technology Documentation.
3. ADXL345 Datasheet: Технічна специфікація [Електронний ресурс]. – Режим доступу: Microchip Technology Documentation.
4. DS1307 Datasheet: Технічна специфікація [Електронний ресурс]. – Режим доступу: Microchip Technology Documentation.
5. ATmega328 Datasheet: Технічна специфікація та опис архітектури мікроконтролера ATmega328. [Електронний ресурс]. – Режим доступу: Microchip Technology Documentation.
6. Індикація на базі HD44780: Методи підключення та керування алфавітно-цифровими РК-дисплеями 16x2. [Електронний ресурс]. – Режими доступу: AllAboutCircuits, DavideGironi Library.
7. White E. Створення вбудованих систем: патерни проектування якісного програмного забезпечення. – O’Reilly Media, 2-ге вид., 2024. – 425 с.
8. Barrett S. F., Pack D. Основи мікроконтролерів Atmel AVR: програмування та інтерфейси. — Morgan & Claypool, 2007. – 180 с.
9. Barnett R. H., Cox S. A., O’Cull L. Програмування вбудованих систем на мові C для контролерів Atmel AVR. – Cengage Learning, 2-ге вид., 2006. – 560 с.
10. Gadre D. V. Програмування та налаштування мікроконтролерів родини AVR. — McGraw-Hill Education, 2000. – 336 с.
11. Van Dam B. Системотехніка мікроконтролерів: 45 прикладних проектів для архітектур PIC, AVR та ARM. – Elektor Publishing, 2009. – 330 с.

ДОДАТКИ

Додаток 1

Лістинг програмного коду, що реалізує головний алгоритм роботи трекера сонячної електростанції для автономних блокпостів поліції

```
// Трекер сонячної електростанції для автономних блокпостів поліції
#include <math.h>
#include <LiquidCrystal.h>
#include <Wire.h>  // для ADXL345
LiquidCrystal lcd(13, 12, 6, 5, 4, 3);  // LCD= RS E D4 D5 D6 D7
#define DEVICE (0x53) // Адрес ADXL345 на шині I2C
#define ADXL345_REG_DEVID         (0x00)  // Device ID має значення 0xE5 (only read)
#define ADXL345_REG_THRESH_TAP    (0x1D)  // коефіцієнт 62,5 mg/LSB
#define ADXL345_REG_OFSX          (0x1E)  // X-axis offset коефіцієнт 15,6 mg/LSB)
#define ADXL345_REG_OFSY          (0x1F)      // Y-axis offset
#define ADXL345_REG_OFSZ          (0x20)      // Z-axis offset
#define ADXL345_REG_BW_RATE       (0x2C)    // Data rate/power mode control – 400 Гц
#define ADXL345_REG_POWER_CTL     (0x2D)      // DATAX, DATAY, DATAZ
#define ADXL345_REG_DATA_FORMAT   (0x31)      // Data format control FULL_RES
#define ADXL345_REG_DATAX0            (0x32)  // X-axis data 0
#define ADXL345_REG_DATAX1            (0x33)  // X-axis data 1
#define ADXL345_REG_DATAY0            (0x34)  // Y-axis data 0
#define ADXL345_REG_DATAY1            (0x35)  // Y-axis data 1
#define ADXL345_REG_DATAZ0            (0x36)  // Z-axis data 0
#define ADXL345_REG_DATAZ1            (0x37)  // Z-axis data 1
#define ADXL345_REG_FIFO_CTL          (0x38)  // FIFO control
#define ADXL345_REG_FIFO_STATUS       (0x39)  // FIFO status
#define ADXL345_BW_800  0xE // 1110 - 1600 - швидкість передачі даних
#define ADXL345_BW_400  0xD // 1101 -  800
#define ADXL345_BW_200  0xC // 1100 -  400
#define ADXL345_BW_100  0xB // 1011 -  200
//------------------- end define ADXL345--------------------------
#define DS1307 0xD0             // адрес на шині I2C для DS1307 RTC
#define ROM24C32  0xA0          // адрес на шині I2C для 24C32
#define TW_START 0xA4           // відправка start condition (TWINT,TWSTA,TWEN)
#define TW_READY (TWCR & 0x80) // TWI в стані готовності (TWINT повернувся в  лог. 1)
#define TW_STATUS (TWSR & 0xF8)   // повертає значення статуса
#define TW_STOP 0x94              // послати stop condition (TWINT,TWSTO,TWEN)
#define I2C_Stop() TWCR = TW_STOP   // inline-макрос для stop condition
#define TW_NACK 0x84      // читання даних з NACK ( означає, що це останній байт)
#define READ 1
#define TW_SEND 0x84          // команда відправки даних (TWINT,TWEN)
#define SECONDS_REGISTER   0x00  //  регістр секунд
#define MINUTES_REGISTER   0x01  //  регістр хвилин
#define HOURS_REGISTER     0x02  //  регістр годин
#define DAYOFWK_REGISTER   0x03  //  регістр день тижня  (1=Sunday, 7=Saturday)
#define DAYS_REGISTER      0x04  //  регістр  день місяця  (1 to 31)
#define MONTHS_REGISTER    0x05  //  регістр  місяць
#define YEARS_REGISTER     0x06  //  регістр  рік   (0 to 99)
#define CONTROL_RG         0x07  //  регістр  управління(0X10 - SQ - імпульси 1Гц)
```

```
#define F_CPU 16000000L // Частота  CPU рівна 16 МГц
#define F_SCL 100000L   // Частота I2C 100 кГц
// Адреси регістрів AS5600 та I2C адреса
// ============================================================================
#define AS5600_I2C_ADDRESS 0x36 // Стандартна I2C адреса AS5600 – 6D, 6C
#define AS5600_REG_RAW_ANGLE_MSB 0x0C // Старший байт сирого кута (bits 11:8)
#define AS5600_REG_RAW_ANGLE_LSB 0x0D // Молодший байт сирого кута (bits 7:0)
// Регістри для зчитування відфільтрованого кута (Filtered Angle)
#define AS5600_REG_ANGLE_MSB 0x0E // Старший байт відфільтрованого кута (bits 11:8)
#define AS5600_REG_ANGLE_LSB 0x0F // Молодший байт відфільтрованого кута (bits 7:0)
// Регістри конфігурації (CONFIG)
#define AS5600_REG_ZMCO 0x00      // Zero Position Coarse (bits 1:0)
#define AS5600_REG_ZPOS_MSB 0x01  // Zero Position MSB (bits 7:0)
#define AS5600_REG_ZPOS_LSB 0x02  // Zero Position LSB (bits 7:0)
#define AS5600_REG_MPOS_MSB 0x03  // Maximum Position MSB (bits 7:0)
#define AS5600_REG_MPOS_LSB 0x04  // Maximum Position LSB (bits 7:0)
#define AS5600_REG_MANG_MSB 0x05  // Maximum Angle MSB (bits 7:0)
#define AS5600_REG_MANG_LSB 0x06  // Maximum Angle LSB (bits 7:0)
#define AS5600_REG_CONF_MSB 0x07  // Configuration MSB (bits 7:0)
#define AS5600_REG_CONF_LSB 0x08  // Configuration LSB (bits 7:0)
// Регістри статусу (STATUS)
#define AS5600_REG_STATUS 0x0B        // Статус (MD, ML, MH біти)
#define AS5600_REG_AGC 0x1A           // Автоматичне керування підсиленням (AGC)
#define AS5600_REG_MAGNITUDE_MSB 0x1B  // Сила магнітного поля MSB
#define AS5600_REG_MAGNITUDE_LSB 0x1C  // Сила магнітного поля LSB
//------------------------------------------------------------- adxl345----
float range_X;    // Scale Factor / Gain;
float range_Y;    // Scale Factor / Gain;
float range_Z;    //Scale Factor / Gain;
float scale_X;  // Коефіцієнт масштабу (фактична чутливість в LSB/g  для осі X)
float scale_Y;  // це фактична чутливість в LSB/g для осі Y
float scale_Z;  // це фактична чутливість в LSB/g для осі Z
float gain_X;  // коефіцієнт підсилення по осі X
float gain_Y;  // коефіцієнт підсилення по осі Y
float gain_Z;  // коефіцієнт підсилення по осі Z
float etalonxyz = 256.0;
float roll_rad;    // кут Roll (Крен)
float roll_deg;   // Перетворюємо радіани в градуси
float pitch_rad ;  // кут Pitch (Тангаж)
float pitch_deg ; // Перетворюємо радіани в градуси
float offsetX = 0;   // offsetX = (AverageX_pos_g + AverageX_neg_g) / 2
float offsetY = 0;   // OffsetY = (AverageY_pos_g + AverageY_neg_g) / 2
float offsetZ = 0;    // OffsetZ = (AverageZ_pos_g + AverageZ_neg_g) / 2
int AverageX_pos_g = 258;   // виміри для калібрування ADXL345 - підключено!!!
int AverageX_neg_g = -270;  // виміри для калібрування ADXL345 підключено!!!
int AverageY_pos_g = 240;   // виміри для калібрування ADXL345 підключено!!!
int AverageY_neg_g = -291;  // виміри для калібрування ADXL345 - підключено!!!
int AverageZ_pos_g = 224;   // виміри для калібрування ADXL345 - підключено!!!
int AverageZ_neg_g = -282;  // виміри для калібрування ADXL345 - підключено!!!
byte _buff[6];
byte id;
//змінні - для зберігання значень прискорень по осях (додатні і від'ємні значеннння)
```

```
volatile signed int axisX, axisY, axisZ;
volatile signed int X, Y, Z;
float k_fr = 3.9;   // коефіцієнт  вихідних даних 3,9 mg/LSB,
volatile float xa = 0.0, ya = 0.0, za = 0.0; //реальне прискорення по осі x (м/с2)
float   axyz = 0; // кути відхилення платформи від початкового положення
//Вибір роздільної здатності ADXL345_REG_DATA_FORMAT
typedef  enum
{
  RESOLUTION_FULL      = 8, // або 0xB = 11 ???
  RESOLUTION_2         = 0,
 } Set_Resol;
//------------------------------------------------------ adxl345----------
const int TH = 60;
//------ для SUN TRACKER ----------------
const int   HS = 3600; // година має секунд
const int   HMS = 60;
const int   GPI05 = 180;
const float NX = 23.45; // схилення
const float KNX = 0.986; // 360/365 = 0.986;
typedef union
{ unsigned int i;
  float f;
} value;
enum {TEMP, HUMI};

value humi_val, temp_val;
char  work = 0;
int iz = 0;                              // duty для пищика
volatile byte  d_day, d_month, d_year, d_wk;   // змінні годинника дата
volatile byte d_hour,  d_minute,  d_second;   // змінні годинника година
byte thour = 18,  tminute = 30,  tsecond = 0;       // змінні таймера вмикання поливу
int timex = 0;      //  реальний час в хвилинах
int timeon = 0;     // час включення термостата в хвилинах
int delta = 10;     // delta= timeon - timex =0 - умова вмикання поливу по таймеру
volatile byte mem_byte = 0;
unsigned int adress;
byte flagc = 0;  // прапорець для виводу дати
//------------ SUN TRACKER -*****************************************
float fi = 50.76;      // широта xxx  в градусах+ =N
int  fihigh, filow ;
float Lambda = 48.75; // довгота xxx  в градусах + =E
int Lambdahigh, Lambdalow ;
int DUAh; // часова зона для перетворень
float EoT = 0;          // рівняння часу в хвилинах (перевести в секунди)
int Neot = 0;         // пройшло днів від початку року
float Beot =0.0, Beotrad =0.0, Beotrad2 =0.0, Beotx =0.0, Beotxrad = 0.0; // кут дня
float KB =360.0, KR =365.0, KRN =365.0, KRV = 366.0; // KRV-рік високосний, KRN - ні!
const byte MASDAY[13] ={0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }; //
const byte VMASDAY[13] =   {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }; //
unsigned char mi;
int rik;
float k1 =9.87, k2 =7.67, k3 =78.7, s1 =0.0, s2 =0.0, s3 =0.0;// коеф рівняння часу
```

```
float originalNumber = 123.456; //  змінна - приклад для обчислень
int  integerPart;      // змінна для обчислення цілої частини дробу
float fractionalPart;  // змінна яка отримує дробову частину дробу
float Deltatime; //різниця в часі між UTC(Грінвіч) і середнім місцевим часом
float EEST = 0.0, UTC = 0.0,  LST = 0.0, DUA = 3.0; // UTC - час по Грінвічу  LST  - істинний
Сонячний час (AST) !!! DUA - різниця в часі в Україні літо! Зимою=2
float dgradus = 0.0; //  в градусах схилення Сонця - залежить від дати. NX= 23.45;
float HA;  //  — Hour Angle-годинний кут Сонця . H=(AST−12 годин)15 градусів
float Altitude; // висота Сонця над горизонтом - змінюється в часі.
float rfi, rd, rha;
float HAsunrise = 0.0;   // годинний кут сходу Сонця  на вказану дату
float timesunday = 0.0;  // тривалість сонячного дня на вказану дату
float tsunrise = 0.0, sunrisetimeh=0.0, sunrisetimem=0.0; // час сходу Сонця
float tsunset = 0.0, sunsetimeh=0.0,  sunsetimem =0.0; // час заходу Сонця
float sunculmimationtimeh =0.0, sunculmimationtimem =0.0;
float Aze = 0.0;         // азимут географічний сходу Сонця
float Azw = 0.0;         // азимут географічний заходу Сонця
float Azt = 0.0;         // біжучий азимут географічний
float B, C, D, E, F, G;  // проміжні змінні для обчислень азимуту Сонця
float Am = 0.0; // азимут магнітний
float DM =- 8.4; //магнітне схилення dm =8.4 W то DM=-8.4, якщо dm =8.4 E то DM= +8.4
float G180=180.0, G360=360.0;
byte sunup[8] =   {B00100, B01110, B11111, B00100, B00100, B00100, B00100, B00000 }; //
піктограма стрілка вверх
byte sundown[8] = {B00100, B00100, B00100, B00100, B11111, B01110, B00100, B00000}; //
піктограма стрілка вниз
byte sunzenit[8] = {B00100, B10101, B01110, B11111, B01110, B10101, B00100, B00000}; //
піктограма Сонце
//---------------------- клавіші ******************************************
const int KH1 = 8;      //  на pin.8 під'єднано клавішу "MENU"
const int KH2 = 9;      //  на pin.9 під'єднано клавішу "UP"
const int KH3 = 10;     //  на pin.10 під'єднано клавішу "DOWN"
const int KH4 = 11;     //  на pin.11 під'єднано клавішу "EXIT"

int val1 = HIGH, val2 = HIGH, val3 = HIGH, val4 = HIGH;
int old_val1 = HIGH, old_val2 = HIGH, old_val3 = HIGH, old_val4 = HIGH;
char menu_title[] = "*  Main  Menu  *"; // Заголовок головного меню
//--------------------------------------------------------------------------

// функція читати статус  магнітометра as5600
int16_t as5600_read_status(void)
{
   uint8_t status_byte;
   if (i2c_read_bytes(AS5600_REG_STATUS, &status_byte, 1) != 0) {
      return -1; // Помилка зчитування
   }
   return (int16_t)status_byte;
}  // ---------------------------- end as5600_read_status ------------------
// функція читати значення сили магнітного поля
int16_t as5600_read_magnitude(void) //12-біт значення магнітного поля(0-4095),
{
   uint8_t buffer[2];
```

```
    if (i2c_read_bytes(AS5600_REG_MAGNITUDE_MSB, buffer, 2) != 0) {
        return -1; // Помилка зчитування
    }
    return (int16_t)((buffer[0] << 8) | buffer[1]);
} // ---------- end as5600_read_magnitude ----------

// функція читати значення чутливості до магнітного поля
int16_t as5600_read_agc(void) // Значення AGC (0-255), або -1 у випадку помилки
{
    uint8_t agc_byte;
    if (i2c_read_bytes(AS5600_REG_AGC, &agc_byte, 1) != 0) {
        return -1; // Помилка зчитування
    }
    return (int16_t)agc_byte;
} //---------- end as5600_read_agc----------

// функція читати значення  регістра ZMCO
int16_t as5600_read_zmco(void) //Регістр ZMCO показує,скільки разів ZPOS і MPOS були
записані.
{
    uint8_t zmco_byte;
    if (i2c_read_bytes(AS5600_REG_ZMCO, &zmco_byte, 1) != 0) {
        return -1; // Помилка зчитування
    }
    return (int16_t)zmco_byte;
} //---------- end as5600_read_zmco----------

// функція читати сире значення  кута з AS5600
int16_t as5600_read_raw_angle(void) // Зчитує сире 12-бітне значення кута з AS5600
{
    uint8_t buffer[2];
    if (i2c_read_bytes(AS5600_REG_RAW_ANGLE_MSB, buffer, 2) != 0) {
        return -1; // Помилка зчитування
    }
    // AS5600 виводить 12-бітне значення:
    // buffer[1] містить молодші 8 біт (LSB).
    // Повне 12-бітне значення: (MSB << 8) | LSB
    return (int16_t)((buffer[0] << 8) | buffer[1]);
} //---------- end as5600_read_raw_angle ----------
// функція читає відфільтроване значення  кута з AS5600
int16_t as5600_read_angle(void)
{
    uint8_t buffer[2];
    if (i2c_read_bytes(AS5600_REG_ANGLE_MSB, buffer, 2) != 0) {
        return -1; // Помилка зчитування
    }
    return (int16_t)((buffer[0] << 8) | buffer[1]);
} //---------- end as5600_read_angle ----------
//---------- setup-!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
void setup()
{
  Wire.begin();                // ініціалізувати шину i2c
```

```
  // I2C_Init();              // мій I2C_Init()
  lcd.begin(16, 2);
  Serial.begin(9600); // Open serial connection to report values to host

  ADXL345_Init ();
  writeTo(ADXL345_REG_POWER_CTL, 0x00);   // standby mode =0
  delay(20);
// clear any existing offsets
  writeTo(ADXL345_REG_OFSX, 0); // обнулення  OFSX
  writeTo(ADXL345_REG_OFSY, 0); // обнулення  OFSY
  writeTo(ADXL345_REG_OFSZ, 0); // обнулення  OFSZ
  writeTo(ADXL345_REG_FIFO_CTL, 0);              // Bypass - буфер відключено
  writeTo( ADXL345_REG_BW_RATE, ADXL345_BW_200);  // Buffer Speed - 400Hz
  writeTo(ADXL345_REG_DATA_FORMAT, 0x00);        //  resolution, +/-2g, 3,9 mg/LSB.
  offsetX = (AverageX_pos_g + AverageX_neg_g) / 2; // зміщення по осі x
  offsetY = (AverageY_pos_g + AverageY_neg_g) / 2; // зміщення по осі y
  offsetZ = (AverageZ_pos_g + AverageZ_neg_g) / 2; // зміщення по осі z
  range_X = AverageX_pos_g - AverageX_neg_g;  // діапазон вимірювання (сирий) по X
  range_Y = AverageY_pos_g - AverageY_neg_g;  // діапазон вимірювання (сирий) по Y
  range_Z = AverageZ_pos_g - AverageZ_neg_g;  // діапазон вимірювання (сирий) по Z
  scale_X = range_X / 2;  // (це фактична чутливість в LSB/g для осі X)
  scale_Y = range_Y / 2;  // (це фактична чутливість в LSB/g для осі Y)
  scale_Z = range_Z / 2;  // (це фактична чутливість в LSB/g для осі Z)
  gain_X = etalonxyz / scale_X; // коригуючий множник
  gain_Y = etalonxyz / scale_Y; // коригуючий множник
  gain_Z = etalonxyz / scale_Z; // коригуючий множник
  writeTo(ADXL345_REG_POWER_CTL, 0x08);    //  Measure =1
  delay(20);
  readFrom(ADXL345_REG_DEVID, 1, _buff); // читати id - один байт в буфер
  id = _buff[0];
  if (id != 0xE5) // Check connection якщо не прочитано ID=0xE5 то "помилка!"
  {
   Serial.println("ADXL345 error "); // вивід в послідовний порт інф. про помилку
   lcd.setCursor(2, 0);
   lcd.print("ADXL345 error ");
   delay(500);
  }
  else
  {
   Serial.println("  ");     Serial.print("ADXL345 ID: ");
   Serial.println( id, HEX ); // вивести ID в  послідовний порт
   lcd.setCursor(0, 0);
   delay(300);
  };
// load characters to the LCD
  lcd.createChar(0, sunup); // записати символ "стрілка вверх" в пам'ять LCD
  lcd.createChar(1, sundown);   // записати символ "стрілка вниз" в пам'ять LCD
  lcd.createChar(2, sunzenit);  // записати створений символ  "Сонце" в пам'ять LCD
  delay(10);  lcd.clear();  delay(10);
  lcd.print("* SUN TRACKER * ");  // Print a message to the LCD.
  delay(1000);   lcd.setCursor(0, 0);  lcd.print("               ");
  delay(20);
```

```
  //------------------------- set time & date -**********************************
  d_day = 10; d_month = 4; d_year = 26; d_wk = 5; // змінні годинника - дата
  d_hour = 11;  d_minute = 10;  d_second = 00;   // змінні годинника - година
SetDate( d_day, d_month, d_year, d_wk);  // 09 02 2024 th -для запуску годинника
 SetTime(d_hour, d_minute, d_second);     // 11:41:52
//readDATE();   delay(10);   // отримати дату
// readTIME();   delay(10);   // отримати час
  //----------------- end set time & date ------------------------------------
  I2C_WriteRegister(CONTROL_RG,   0x10);  // виводити  на pin.7 імпульси частотою 1Гц
  delay(100);
  pinMode(KH1, INPUT);  //  KH1 is an input - "MENU"
  pinMode(KH2, INPUT);  //  KH2 is an input  - "Select+"
  pinMode(KH3, INPUT);  //  KH3 is an input  - "Select-"
  pinMode(KH4, INPUT);  //  KH4 is an input  - "EXIT"
  pinMode(KH1, INPUT_PULLUP);  // PULLUP ДО +5В
  pinMode(KH2, INPUT_PULLUP);  // PULLUP ДО +5В
  pinMode(KH3, INPUT_PULLUP);  // PULLUP ДО +5В
  pinMode(KH4, INPUT_PULLUP);  // PULLUP ДО +5В
}
// ********************************** LOOP ##################################
void loop()
{
byte error, address;
delay(100);         // затримка
  Get_Data (30);  // 30- кількість вимірів для обчислення середнього значення
  xa = (axisX - offsetX) * gain_X;
  xa = (xa * k_fr) / 100;           // обчислення прискорення по осі x - *9.81 м/с2
  ya = (axisY - offsetY) * gain_Y;
  ya = (ya * k_fr) / 100;           // обчислення прискорення по осі y - *9.81 м/с2
  za = (axisZ - offsetZ) * gain_Z;
  za = (za * k_fr) / 100;          // обчислення прискорення по осі z - *9.81 м/с2

//*****************************************************************************
  // Обчислюємо кут Roll (Крен)
  roll_rad = atan2(ya, za);
  roll_deg = roll_rad * 180.0 / PI; // Перетворюємо радіани в градуси
  // Обчислюємо кут Pitch (Тангаж)
  pitch_rad = atan2(-xa, sqrt(ya * ya + za * za));
  pitch_deg = pitch_rad * 180.0 / PI; // Перетворюємо радіани в градуси
  // put your main code here, to run repeatedly:
  readDATE();   delay(10);   // отримати дату
  readTIME();   delay(10);   // отримати час
   prntimedatelcd();  // вивід дати і часу на LCD
  //------------------------- read KH  в циклі*******************************
  val1 = digitalRead(KH1);               // читати значення клавіші
  delay(50);
  if ((val1 ==LOW ) && (old_val1 ==HIGH )) // аналіз  натискань клавіші
  {
    Serial.println(" MENU ");              // натиснута клавіша "MENU"
    Serial.println();
    mainMenu() ;
  };
```

```
old_val1 = val1;                    // val is now old, let's store it
delay(50);
val4 = digitalRead(KH4);                // читати значення клавіші KH4  EXIT
delay(10);
if ((val4 ==LOW ) && (old_val4 ==HIGH ))  // аналіз  натискань "EXIT"
{
  Serial.print(" *************   SETUP INFORMATION   ******************** ");  Serial.println(" ");
  Serial.println(" ");
  Serial.print("  TIME ZONE = +"); Serial.print(DUA,0); Serial.print(" h"); //
  Serial.println(" ");
  Serial.println(" ");
 // Serial.print("  ****************************************** ");  Serial.println(" ");
};
old_val4 = val4; // val is now old, let's store it
delay(50);          // затримка

//------------------------end kh info ---------------------------------
// ---------------------- SUN TRACKER -*******************************************
prntimedatelcd();  // вивід дати і часу на LCD
lcd.setCursor(9, 0); lcd.print(fi); lcd.print((char)223);      // вивід широти
if (fi >= 0) {     lcd.print("N");   }
else { lcd.print("S");   };   // N - північна широта, S -південна
lcd.setCursor(9, 1); lcd.print(Lambda); lcd.print((char)223); // вивід довготи
if (Lambda >= 0) {     lcd.print("E");   }
else {
  lcd.print("W");
};      // E - східна довгота, W -західна від Грінвіча
Serial.println(" ********************************************************** ");
Serial.print(" LOCATION : ");
Serial.print(fi); Serial.print(" degrees N     ");
Serial.print(Lambda); Serial.println(" degrees E   ");
Serial.println(" ");
printOut();  // вивід значення  часу  в Україні ----------------
  Serial.print("  day =");    Serial.print(d_day);
Serial.print("  month ="); Serial.print(d_month);
Serial.print("  year =20");  Serial.println(d_year);
//------- обчислення рівняння часу EOT -*********************************
rik = d_year % 4; // 0- високосний рік
if (rik == 0) {
  KR = KRV;
} else {
  KR = KRN;
}; // KRV = 366.0;- високосний рік, KRN=365 днів
if (rik == 0)
{
  Neot = 0;
  for (mi = 0; mi < d_month; mi++)
  {
    Neot = Neot + VMASDAY[mi];
  };
  Neot = Neot + d_day; // минуло днів від початку року
}
```

```
  else
  {
    Neot = 0;
    for (mi = 0; mi < d_month; mi++)
    {
      Neot = Neot + MASDAY[mi];
    };
    Neot = Neot + d_day; // минуло днів від початку року
  };

  Beot = (KB / KR) * (Neot - 81) ; //  в градусах
  Beotrad = (Beot * PI) / GPI05;     //  в градусах  const int   GPI05 = 180;
  Beotrad2 = Beotrad * 2;          //  в градусах
  s1 = sin(Beotrad2);
  s3 = ((Beot + k3) * PI) / GPI05; //  k3=78.7
  s2 = sin(s3);
  EoT = (k1 * s1) - (k2 * s2); // у хвилинах (в межах -14.. +16,5 хвилин)
  //  k1= 9.87, k2=7.67 коефіцієнти для обчислення рівняння часу
  originalNumber = EoT; // переведення дробової частини в секунди
  integerPart = (int)originalNumber; // Отримуємо цілу частину: 123
  fractionalPart = originalNumber - integerPart; // 123.456 - 123 = 0.456
  fractionalPart = fractionalPart * HMS;
    //-------------------------------------------------------------------------
  // ------ обчислення різниці в часі між UTC і LST -**************************
  Deltatime = Lambda * 4; // у хвилинах
  originalNumber = Deltatime / HMS ;
  integerPart = (int)originalNumber;
    fractionalPart = (originalNumber - integerPart) * HMS;
    Serial.println("  ");
  EEST = float(d_hour) * HS + float(d_minute) * HMS + float(d_second);
  EEST = EEST / HMS; // у хвилинах
  Serial.print(" EEST = ");  // вивести час в Україні
  Serial.print(EEST, 2); Serial.print(" min   ");
  Serial.print(d_hour); Serial.print(" h "); Serial.print(d_minute); Serial.print(" min  ");
Serial.print(d_second); Serial.print(" s   ");
  Serial.print("  UKRAINE ");
  Serial.println("  ");
  UTC = float(d_hour) * HS + float(d_minute) * HMS + float(d_second) - float(DUA * HS) ; // у
секундах
  UTC = UTC / HMS; //  час по Грінвічу у хвилинах
  originalNumber = UTC / HMS ; // переведення UTC у години
  integerPart = (int)originalNumber;
  Serial.print(" UTC =  ");
  Serial.print(UTC, 2); Serial.print(" min   ");
  Serial.print(integerPart); Serial.print(" h "); // години
  fractionalPart = (originalNumber - integerPart) * HMS;
  Serial.print(fractionalPart); Serial.print(" min ");   //Greenwich
  Serial.print("      Greenwich ");   Serial.println("  ");
  LST = UTC + Deltatime;    // Місцевий Середній Сонячний Час (LMST) -
    originalNumber = LST / HMS ; //переведення LST у години
  integerPart = (int)originalNumber;
    fractionalPart = (originalNumber - integerPart) * HMS;
```

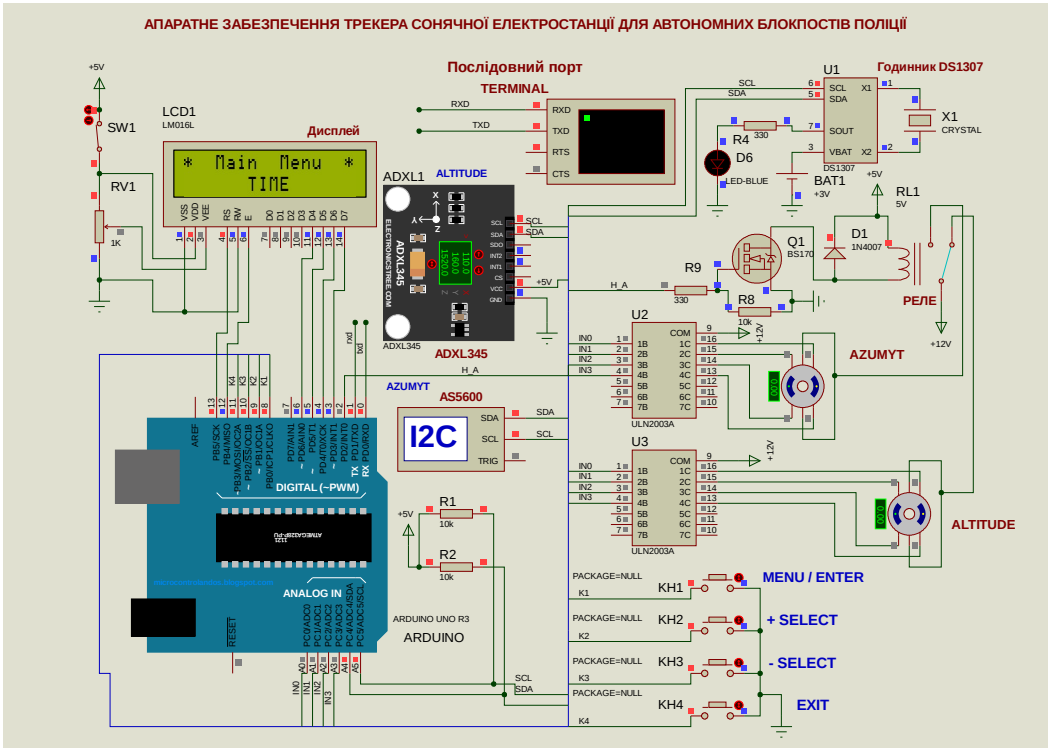
```
  LST = UTC + Deltatime + EoT; // Істинний Сонячний Час (AST)
Serial.print(" AST =  "); Serial.print(LST, 2); Serial.print(" min   ");
originalNumber = LST / HMS ; //переведення LST у години
integerPart = (int)originalNumber;
Serial.print(integerPart); Serial.print(" h "); // години
fractionalPart = (originalNumber - integerPart) * HMS;
Serial.print(fractionalPart); Serial.print(" min ");
Serial.print("    True solar time! ");
Serial.println("  ");
Serial.println("  ");
//---------------- обчислення схилення Сонця ************************
originalNumber = float(Neot) - 81;
originalNumber = originalNumber * KNX * PI / GPI05; //  в радіанах
dgradus = NX * sin(originalNumber); // в градусах  // схилення в градусах
 //-------------------------------------------------------------------------
// ---------- обчислення годинного кута -*************************
HA = (LST - 12 * HMS) * 15 / HMS; // в градусах HMS=60
  //-----------------------------------------------------------
// ---------- обчислення висоти Сонця над горизонтом **********************
rfi = fi * PI / GPI05;
rd = dgradus * PI / GPI05;
rha = HA * PI / GPI05;
Altitude = sin(rfi) * sin(rd) + cos(rfi) * cos(rd) * cos(rha);
Altitude = asin(Altitude) / PI * GPI05;
  //------------------- end Altitude
//---------------- обчислення годинного кута сходу Сонця -*********************
HAsunrise = -(tan(rfi) * tan(rd));        // cos(H)=−tan(ϕ)·tan(δ)
HAsunrise = acos(HAsunrise) / PI * GPI05;  // кут сходу Сонця в градусах
HAsunrise = HAsunrise / 15;       //  годинний кут сходу Сонця в годинах
originalNumber = HAsunrise;
integerPart = (int)originalNumber;
  fractionalPart = (originalNumber - integerPart) * HMS;
  //---------------------- end HAsunrise --------------------------------
//---------------- обчислення тривалості дня і часу сходу-заходу Сонця ********
timesunday = HAsunrise * 2;   // тривалість дня
originalNumber = timesunday ; // переведення LST у години
integerPart = (int)originalNumber;
  fractionalPart = (originalNumber - integerPart) * HMS;
  //----------------end timesunday----------------------------
//---------------- час кульмінації Сонця на вказану дату HMS=60 *************
tsunrise = (12 * HMS + (EEST - LST)) / HMS; //
originalNumber = tsunrise ; //переведення LST у години
integerPart = (int)originalNumber;
sunculmimationtimeh=integerPart;    // час кульмінації Сонця в годинах
Serial.print(" Sun culmination   =          ");
Serial.print(sunculmimationtimeh,0); Serial.print(" h "); // години
fractionalPart = (originalNumber - integerPart) * HMS;
sunculmimationtimem=fractionalPart;      // час кульмінації Сонця в хвилинах
Serial.print(sunculmimationtimem,0); Serial.print(" min ");
Serial.print("      EEST (UKRAINE) ");
Serial.println("  ");
//-------------------------------------------------------------------------
```

```
 // ----------------схід Сонця--**********************************
 tsunrise = ((12 * HMS + (EEST - LST)) - HAsunrise * HMS) ; // схід у хвилинах
 Serial.print(" Sunrise    =   "); Serial.print(tsunrise);
 Serial.print(" min      ");
 tsunrise = ((12 * HMS + (EEST - LST)) - HAsunrise * HMS) / HMS ; // у годинах
 originalNumber = tsunrise ; //переведення LST у години
 integerPart = (int)originalNumber;
 sunrisetimeh = integerPart;       // схід Сонця в годинах
  Serial.print(sunrisetimeh,0); Serial.print(" h "); // години схід
 fractionalPart = (originalNumber - integerPart) * HMS;
 sunrisetimem=fractionalPart;
 Serial.print(sunrisetimem); Serial.print(" min ");
 Serial.print("   EEST (UKRAINE) ");
 Serial.println("  ");
 //--------------------- end tsunrise ------------------------
 // ----------------захід Сонця-**********************************
 tsunset = ((12 * HMS + (EEST - LST)) + HAsunrise * HMS); // захід у хвилинах
 Serial.print(" Sunset     =  ");  Serial.print(tsunset); Serial.print(" min     ");
 tsunset = ((12 * HMS + (EEST - LST)) + HAsunrise * HMS) / HMS ; // у годинах
 originalNumber = tsunset ; //переведення LST у години
 integerPart = (int)originalNumber;
 sunsetimeh = integerPart;       // захід Сонця в годинах
  Serial.print(sunsetimeh,0); Serial.print(" h "); // години схід
 fractionalPart = (originalNumber - integerPart) * HMS;
 sunsetimem =fractionalPart;
 Serial.print(sunsetimem); Serial.print(" min ");
 Serial.print("    EEST (UKRAINE) ");
 Serial.println("  ");
 //--------------------- end tsunset ------------------------------------------
 // ------ азимут сходу- заходу Сонця ***************************
 // cos (Az)= sin(dgradus)/cos(fi) тоді Az= arcos(sin(dgradus)/cos(fi))- в радіанах
 Aze = acos(  sin( dgradus * PI / GPI05) / cos(fi * PI / GPI05)); // радіанах
 Aze = Aze * GPI05 / PI; // в градусах- азимут географічний сходу Сонця
 Serial.println("  ");
 Serial.print(" Az Sunrise =    "); Serial.print(Aze); Serial.print(" degrees     "); Serial.println("  ");
 Azw = GPI05 * 2 - Aze; // азимут географічний заходу Сонця
 Serial.print(" Az Sunset  =   "); Serial.print(Azw); Serial.print(" degrees     "); Serial.println("  ");
 //--------------- end Aze & Azw -----------------------------------------
 //---------------------- біжучий азимут Сонця ***********************
 B = cos(dgradus * PI / GPI05) * sin(HA * PI / GPI05);
 C = cos(Altitude * PI / GPI05);
 D = -(B / C);                       //  D=SIN(Azt)
 Serial.println("  ");
 //  Serial.print(" SIN(Azt) =     "); Serial.print(D,4);Serial.println(" radian     ");
 E = sin(fi * PI / GPI05) * sin(Altitude * PI / GPI05);
 F = cos(fi * PI / GPI05) * cos(Altitude * PI / GPI05);
 G = (sin(dgradus * PI / GPI05) - E) / F; //  G=COS(Azt)
if ((D>=0)&&(G>=0)) {Azt = atan(D / G) * GPI05/PI;};// якщо sin(A) s cos(A)- додатні
 if ((D>=0)&&(G<0)) {Azt = atan(D / G) * GPI05 / PI +G180;};
//якщо sin(A)-додатній, cos(A) - від"ємний
 if ((D<0)&&(G<0)) {Azt = atan(D / G) * GPI05 / PI +G180;};
// якщо sin(A) -від"ємний, cos(A) - від"ємний
```

```
 if ((D<0)&&(G>=0)) {Azt = (atan(D / G) * GPI05 / PI)+G360;};
// якщо sin(A) -від"ємний, cos(A) -  додатній
  Serial.print(" Azt      =    ");
Serial.print(Azt, 2); Serial.print(" degrees     ");
  Serial.println("    True azimuth ! ");
  //------------------ end Azt
  Serial.print(" Altitude =    ");
  Serial.print(Altitude, 2); Serial.print(" degrees     "); Serial.print("    True Altitude ! ");
  Serial.println("  ");  Serial.println("  ");   Serial.println("  ");
    delay(3000);  // час індикації інформації на LCD
  // вивід на LCD часу сходу Сонця ****************************************
  lcd.setCursor(0, 0); lcd.print("               ");
  lcd.setCursor(0, 0);
  lcd.print(char(0)); lcd.print(" ");      // вивід  значка "стрілка вверх"
  if (sunrisetimeh< 10) {lcd.print("0");} lcd.print (sunrisetimeh,0);lcd.print(":");
  if (sunrisetimem< 10) {lcd.print("0");} lcd.print (sunrisetimem,0);lcd.print("  ");
// вивід на LCD часу заходу Сонця ***************************************
  lcd.print(char(1)); lcd.print(" ");     // вивід  значка "стрілка вниз"
  if (sunsetimeh< 10) {lcd.print("0");} lcd.print (sunsetimeh,0);lcd.print(":");  //
  if (sunsetimem< 10) {lcd.print("0");} lcd.print (sunsetimem,0);lcd.print(" ");  //
  // вивід на LCD часу кульмінації Сонця *************************
  lcd.setCursor(9, 1);  lcd.print("       ");   lcd.setCursor(9, 1);
  lcd.print(char(2));   lcd.print(" ");      // вивід  значка "Сонце"
  if (sunculmimationtimeh< 10) {lcd.print("0");} lcd.print (sunculmimationtimeh,0);lcd.print(":");  //
час сходу Сонця
  if (sunculmimationtimem< 10) {lcd.print("0");} lcd.print (sunculmimationtimem,0);lcd.print("
");  // час заходу Сонця
  delay(3000);  // // час індикації інформації на LCD
  lcd.clear();
// вивід на LCD азимутів  і алтитуди  Сонця ************************************
lcd.print("Ar=");lcd.print (Aze,0);lcd.print((char)223);lcd.print("  "); //азимут сходу Сонця в
градусах
  lcd.print("As=");lcd.print (Azw,0);lcd.print((char)223);   //азимут сходу Сонця
  lcd.setCursor(0, 1);
  lcd.print("Ht="); lcd.print(Altitude,0); lcd.print((char)223); lcd.print("  ");    // висота Сонця над
горизонтом в градусах
  lcd.print("At="); lcd.print(Azt,0); lcd.print((char)223);//азимут Сонця в градусах
// вивід в послідовний порт  кутів  ROLL і   PITCH
   Serial.print(" Real ALTITUDE = "); pitch_deg= -(pitch_deg*10 +5);  Serial.println(pitch_deg, 1); //
for example
 //  Serial.println(" ********************************************** ");
Wire.beginTransmission(AS5600_I2C_ADDRESS);
 error = Wire.endTransmission();
    int16_t status = as5600_read_status();
    int16_t raw_angle = as5600_read_raw_angle();  // сире значення кута
   float angle_degrees = (float)raw_angle / 4095.0 * 360.0;
   Serial.print(" REAL AZUMYT = ");  Serial.println(angle_degrees, 2); // 2 знаки
  Serial.println(" ********************************************** ");
  delay(3000);  // // час індикації інформації на LCD
  lcd.clear();
}
```

Додаток 2

Моделювання роботи трекера сонячної електростанції для автономних блокпостів поліції



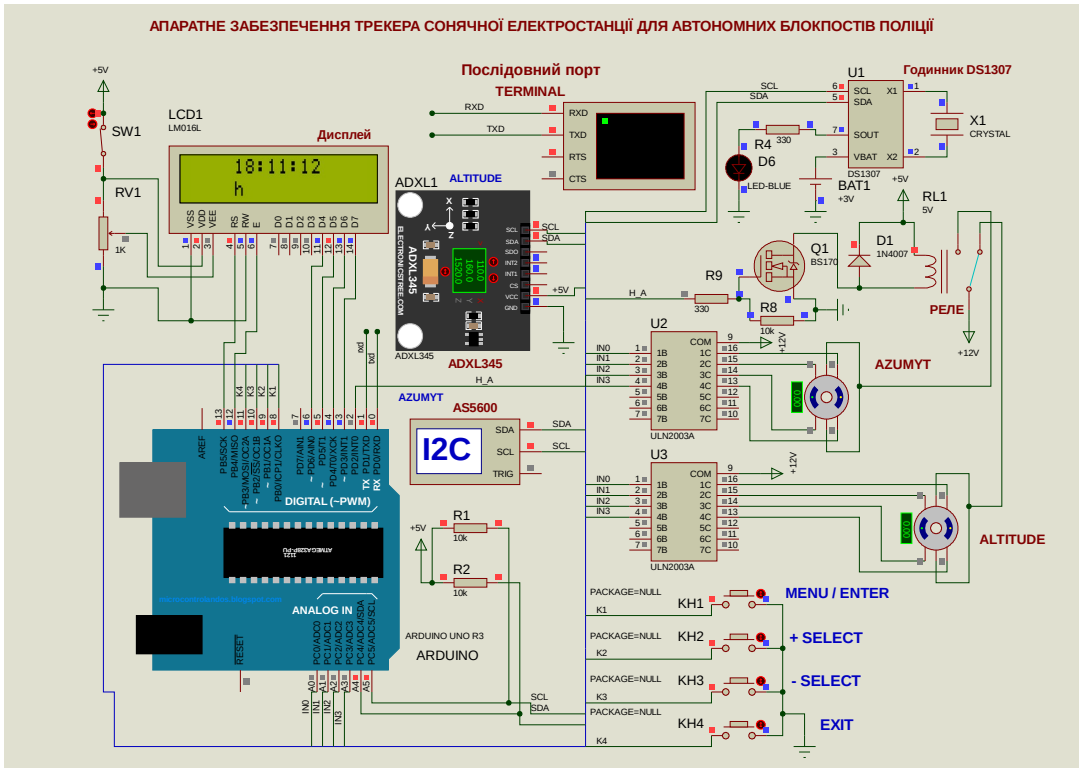


Рис. 1. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (встановлення часу – головне меню)

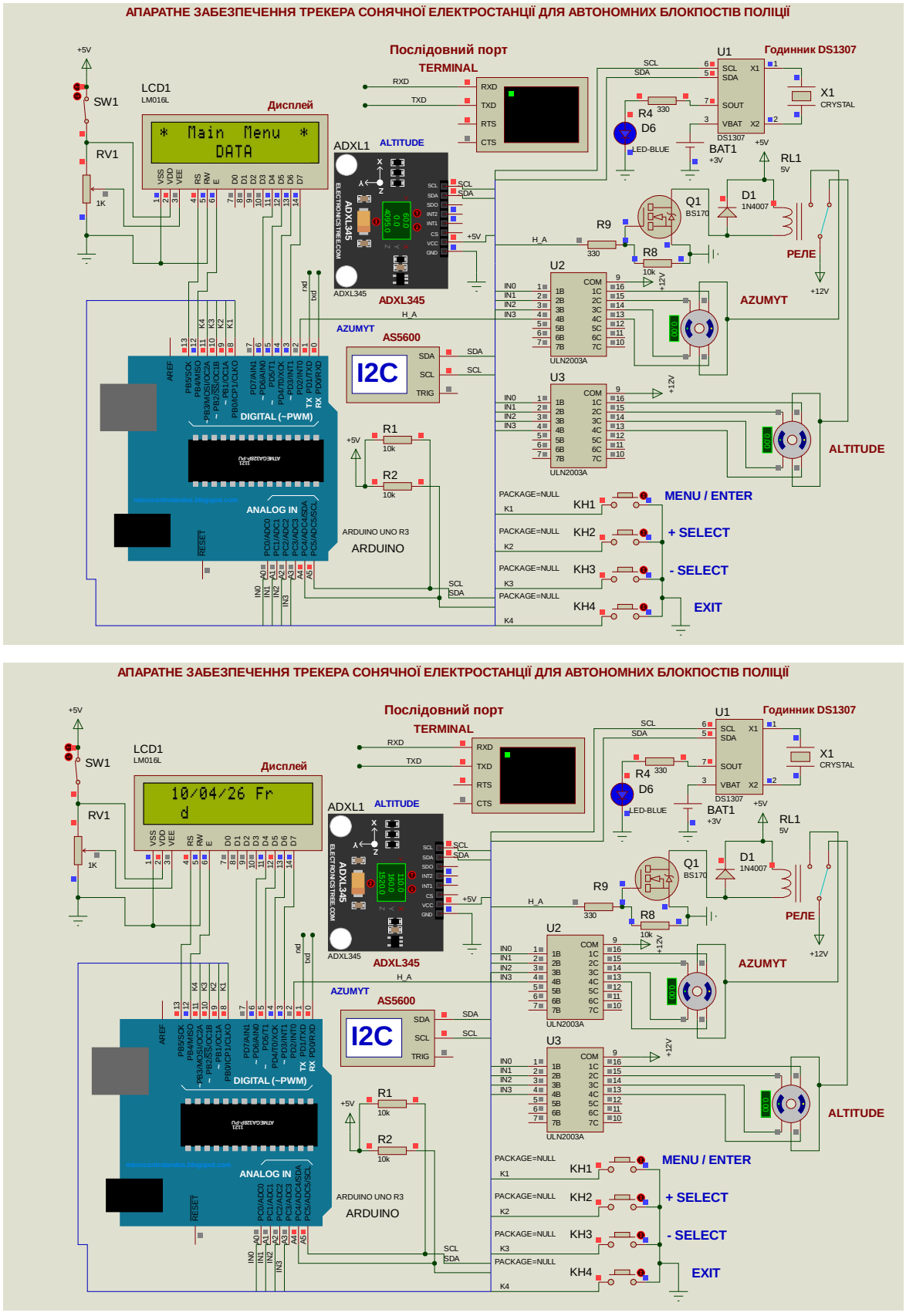


Рис. 2. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (встановлення дати – головне меню)

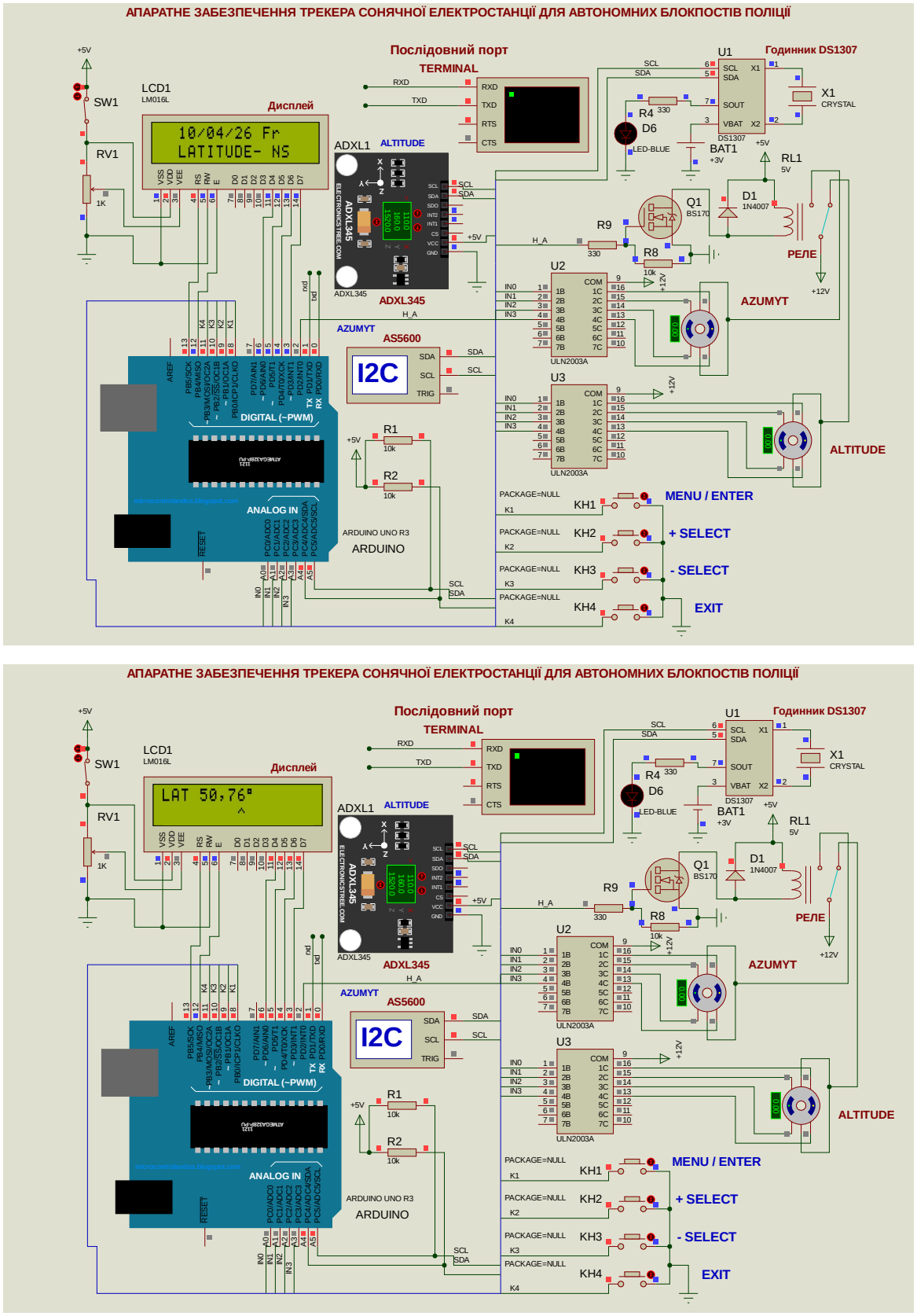
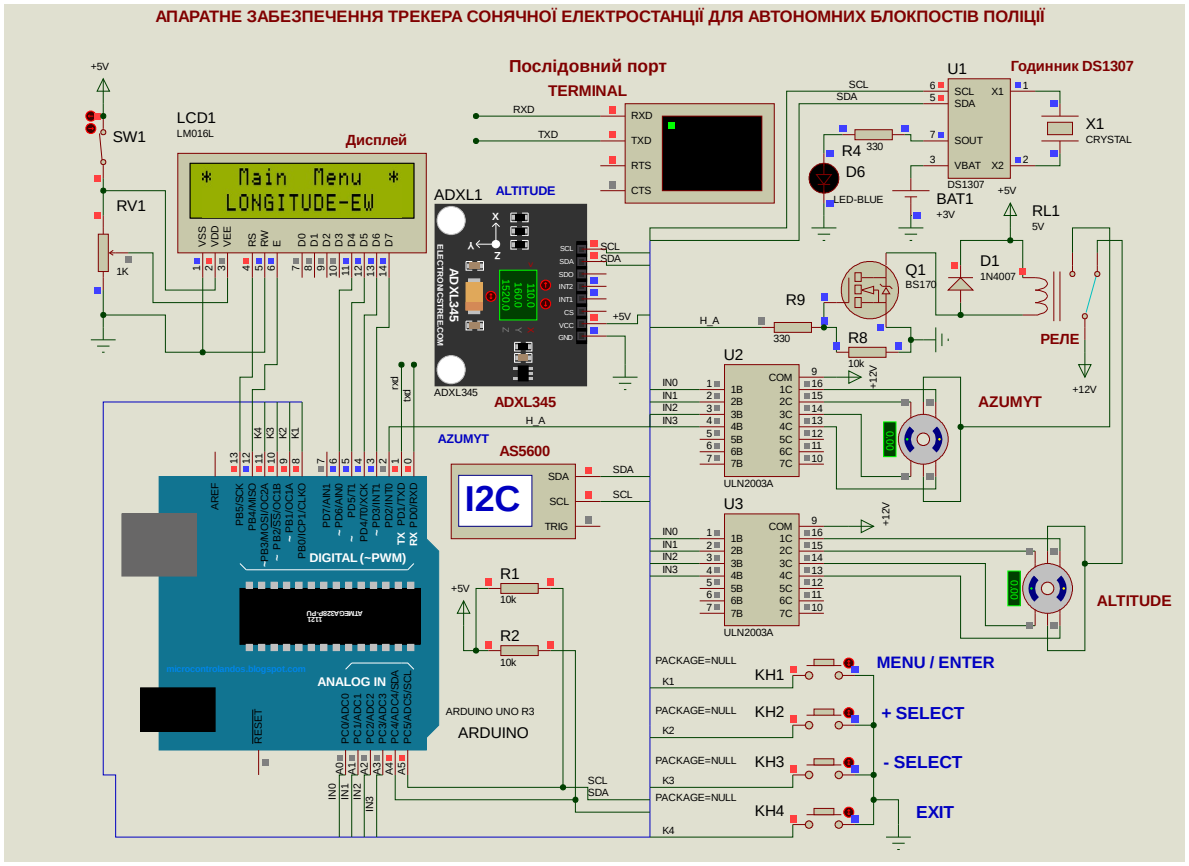


Рис. 3. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (встановлення широти місця – головне меню)



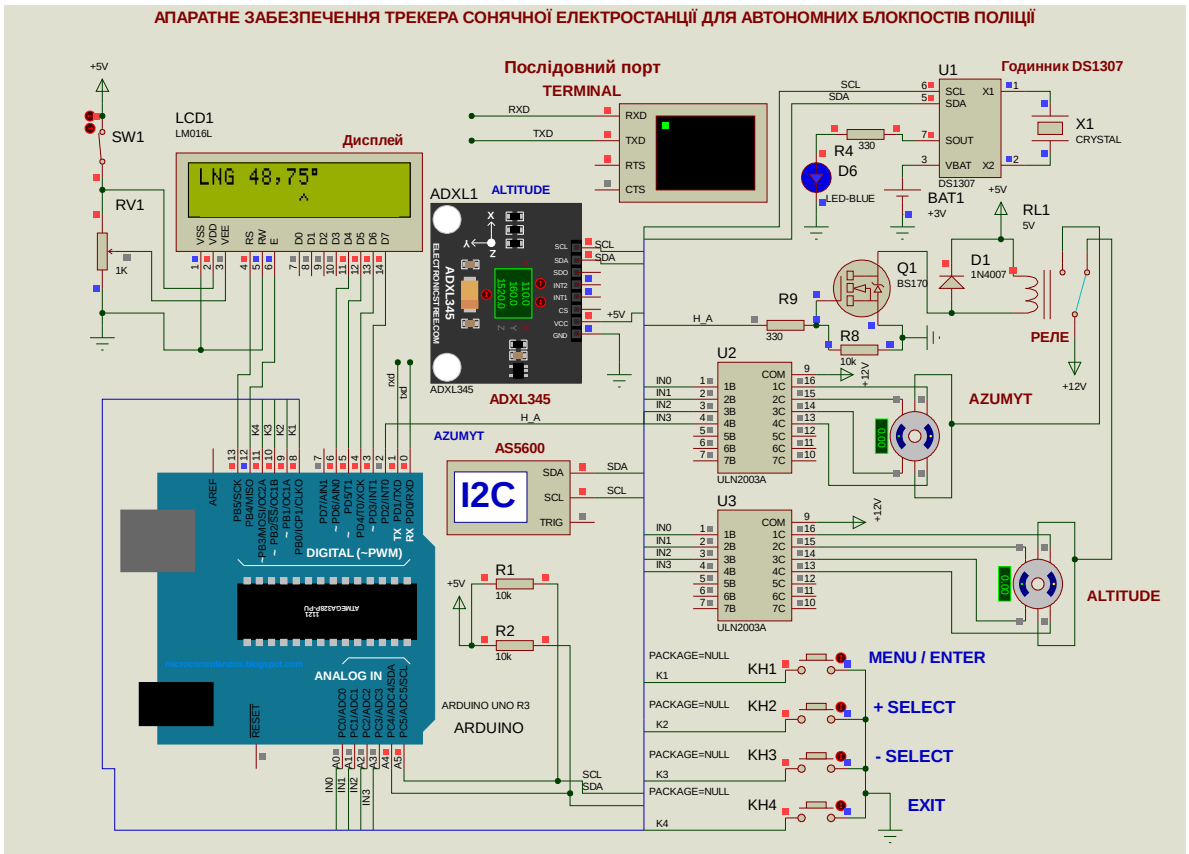


Рис. 4. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (встановлення довготи місця – головне меню)

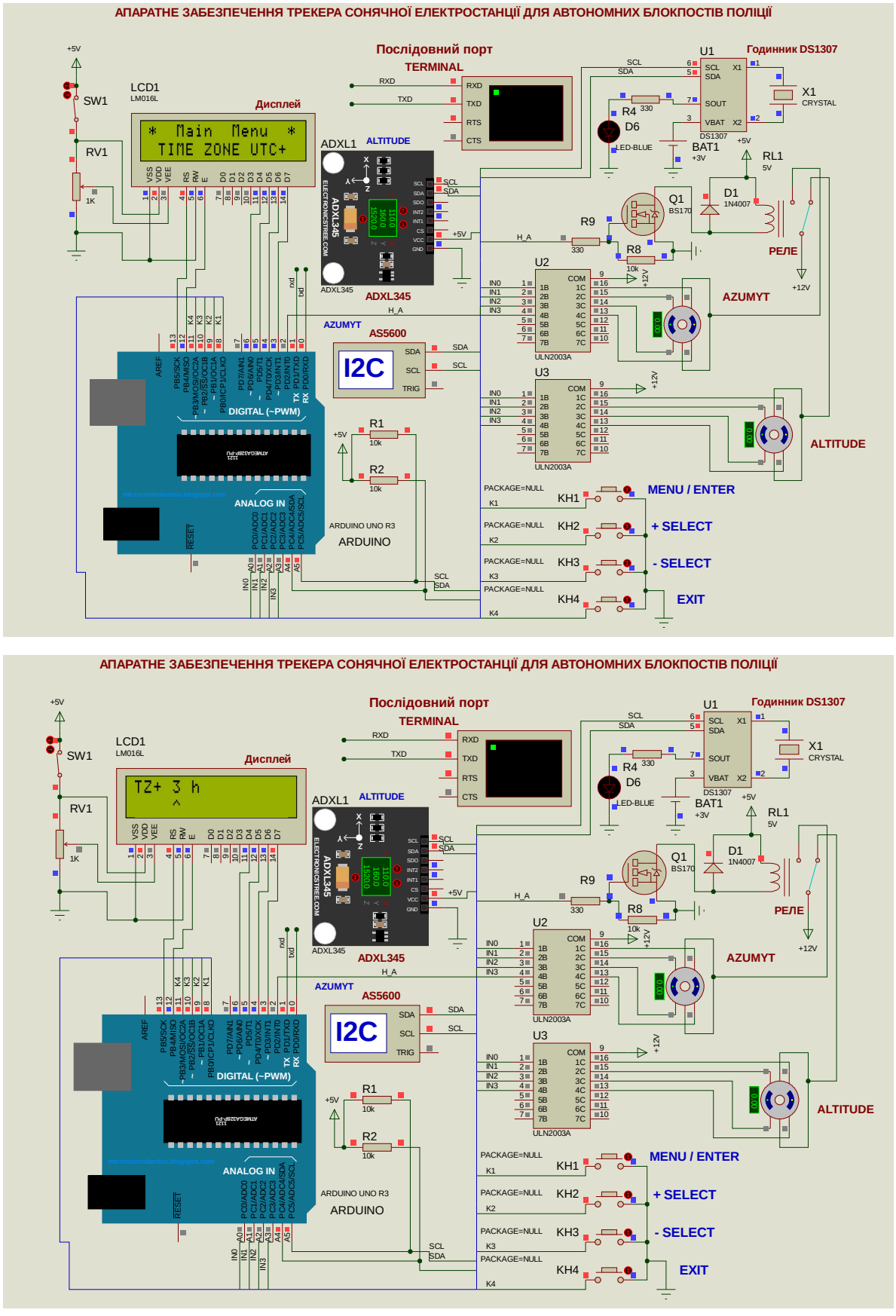


Рис. 5. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (встановлення часової зони – головне меню)

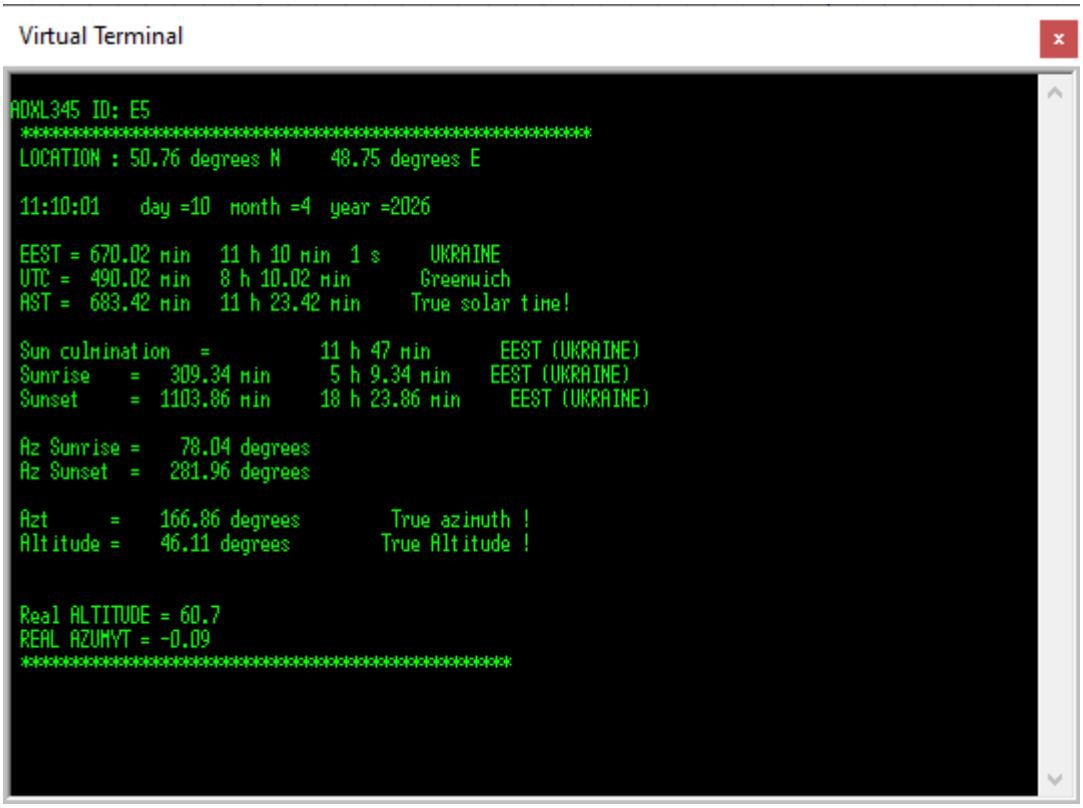


Рис. 6. Моделювання роботи трекера сонячної електростанції в Proteus ISIS (виведення параметрів трекера в послідовний порт – робочий режим)