Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»
Навчально-науковий інститут електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра кіберфізичних та інформаційно-вимірювальних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

здобувача вищої освіти Рульов Євгеній Ігорович
(П.І.Б.)

академічної групи 151-18-1
(шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і назва спеціальності)

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Автоматизація процесів керування сонячною фотоелектричною установкою
(назва за наказом ректора)

Консультанти	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинг.	інституційною	
Керівник кваліфікаційної роботи	ас. Зибалов Д.С.			
Провідний консультант	ас. Зибалов Д.С.			
Розробка апаратного забезпечення системи керування	ст.викл. Проценко С.М.			
Розробка програмного забезпечення системи керування	ст.викл. Бойко О.О.			
Економічна частина	ст. викл. Яремчук І.О.			
Охорона праці	проф. Чеберячко Ю.І.			

Нормоконтролер	ас. Славінський Д.В.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувачем кафедри
кіберфізичних та інформаційно-вимірювальних систем
(повна назва)
_____ Бубліковим А.В.
(підпис) (прізвище, ініціали)

«_____»_____2022 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

здобувача вищої освіти Рульов Є.І. **академічної групи** 151-18-1
(прізвище та ініціали) (шифр)

спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

за освітньо-професійною програмою 151 Автоматизація та комп'ютерно-інтегровані технології
(офіційна назва)

на тему Автоматизація процесів керування сонячною фотоелектричною установкою

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № _____.

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	Вступ. Опис технологічного процесу для об'єкта автоматизації. Огляд існуючих систем автоматизації. Стан питання. Вибір напрямку створення автоматизованої системи.	12.03.2022
Розробка апаратного забезпечення системи керування	Обрання датчиків, виконавчих пристроїв та пристрою керування, розробка структурних схем, функціональної схеми автоматизації та принципової схеми електричної.	02.04.2022
Розробка програмного забезпечення системи керування	Розробка алгоритму керування та програмного забезпечення з людино-машинним інтерфейсом	23.04.2022
Економічна частина	Економічне обґрунтування доцільність витрат на створення системи керування.	07.05.2022
Охорона праці	Розробка організаційно-технічних заходів, щодо реалізації правил безпеки при експлуатації системи.	28.05.2022

Завдання видано _____ ас, Зибалов Д.С.
(підпис п.конс.) (прізвище, ініціали)

Дата видачі 19.01.2022

Дата подання до атестаційної комісії 14.06.2022

Прийнято до виконання _____ Рульов Є.І.
(підпис здобувача) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 127 стор., 51 рис., 23 табл., 3 додат., 20 джерел.

Об’єкт розробки: автоматична система позиціонування фотоелектричних модулів.

Мета: Підвищити кількість генеруємої енергії, зменшити витрати на обслуговування та експлуатацію.

Зазначена актуальність автоматизації, визначений оптимальний спосіб керування. Приведені опис алгоритму функціонування об’єкта. Аналізовані данні нахилу конструкції. Визначенні вхідні та вихідні параметри та розподіл інформаційних потоків. На основі чого був виконаний вибір давачів та виконавчих пристроїв. Обраний однопалатний комп’ютер в якості керуючого пристрою.

Об’єктом управління виступає трекер, поворотний механізм позиціонування. Вихідними параметрами якого слугують: кут нахилу рухомої рами, значення струму та напруги фотоелектричних модулів. Вхідним параметром виступає сигнал керування кроковим двигуном.

Створене відповідне програмне забезпечення. Основною ціллю якого виступає автоматичне управління та безпека роботи установки. А додатковими функціями є: збирання та відображення інформації про стан та значення вихідних параметрів, можливість дистанційного керування та аналізу ефективності реалізуємого об’єкта керування.

СОНЦЕ, ТРЕКЕР, ДИСТАНЦІЙНЕ КЕРУВАННЯ, НАДІЙНІСТЬ, БЕЗПЕКА ЕКПЛУАТАЦІЇ, УПРАВЛІННЯ, ПОЗИЦІОНУВАННЯ, ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЕС – Атомна електростанція

ТЕС – Теплова електростанція

СЕС – Сонячна електростанція

UTC – Всесвітній координований час

ПЗ – Програмне забезпечення

АЦП – Аналого-цифровий перетворювач

ОП – Операційний підсилювач

PID – Ідентифікатор виробника

VID – Ідентифікатор приладу

API – Інтерфейс прикладного програмування

ВСТУП

Сонячна енергія найактивніше розвиваюча галузь енергетики. Останні декілька років у світі спостерігається стрімкий «бум» корпоративних закупок відновлювальної енергії. По даним BlombergNEF за 2019 рік, ринок сонячної енергетики у світі виріс на 44%. За минулий рік було підписано контрактів на будівництво загальною кількість 19.5 ГВт потужностей, а загальна ціна усіх цих проектів перевищує 20 мільярдів доларів.

Існують три основи типи СЕС:

a. Автономні – енергія потрапляє на акумуляторні вузли, а споживачі ізольовані від центральної енергомережі держави.
b. Гібридні – це компромісне рішення яке забезпечує одночасно і автономність і стабільність надання електроенергії. Адже в любий проміжок часу можна підключитися до державної електромережі.
c. Мережеві – вироблена сонячна енергія одразу потрапляє до мережі підприємства, домогосподарства або селища. При цьому не достатня енергія автоматично береться з центральної енергомережі.

Автономні СЕС мають близько 7.4% росту на рік. Гібридні 11.1%, а мережеві 81.5%. Тому є доцільним сконцентруватися на останньому типі, як на найпривабливішому та популярнішому серед замовників.

Перспективність розвитку дуже великі, особливо, с програмами зменшення викидів та забруднення навколишнього середовища. Очікується близько 60% росту у 2022 році, по всьому світу.

Основною проблемою є не ефективне використання фотоелементів для здобування енергії, адже найрозповсюдженим є встановлення на нерухому конструкцію, що знижує ефективність установки через малу кількість часу під час якого промені падають під прямим кутом на поверхню елементу. Тому доцільно розробити систему, яка мінімізує цей фактор, та матиме додаткові конкурентні можливості. Це дозволить збільшити прибутковість та зменшити витрати на обслуговування.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Галузь промисловості

Галузь енергетики є основним утворювачем промисловості та функціонування усіх сфер держави. Адже в сьогодення електропостачання є важливим для населення, сфери послуг, комунікації тощо.

Україна видобуває близько 193,9 млрд кВт/год електроенергії. Більшу частину якої становить АЕС та ТЕС. Останнє сильно впливає на енерго-незалежність України, сировина для електростанцій експортується с закордону. Що слугує загрозою безпеці країни.

З 2014 року Україна активно розвиває альтернативну енергетику, кількість інвестицій у цю сферу за останні 8 років становлять 8.5 млрд євро. Найбільш впливовішим є СЕС. Кількість здобутою енергії збільшилась від 411 МВт у 2014 до 6351 МВт у 4 кварталі 2021р. Окрім промислових електростанцій існують приватні домогосподарства, станом на 01.04.2020 їх кількість становить 24 139 господарств, що добувають та продають державі близько 618 МВт електроенергії. Домінуючим типом електростанцій в Україні становлять системи на базі фотоелектричних модулів.

Причинами, росту є:

a) Екологічність процесу добування електроенергії.
b) Компактність, можливість встановлювати на дахах будинків, виробництв.
c) Простота експлуатації, що дозволяє приватним господарствам встановлювати свої системи енергозабезпечення.
d) Безшумність роботи.
e) Економічність, мала кількість обслуговуючого персоналу та відсутність логістичних, сировинних затрат.
f) Відновлюваність енергії.

Основною метою розвитку є збільшення ефективності здобування електроенергії, зменшення займаних площ та фінансових витрат.

1.2 Технологічний процес

Сонячна електростанція це інженерна споруда яка перетворює сонячну радіацію у електричну енергію. Технологічний процес залежить від типу обраної установки. Розрізняють такі категорії:

a) Парові. Бувають тарілчастого, параболічного або баштового типу. Принци роботи полягає у нагріві води до утворення водяної пари з подальшим проходженням через парову турбіну.
b) Механічні. Концентратор фокусує сонячне світло на теплоприймач двигуна Стерлінга, який починає виконувати механічну роботу, завдяки розширенню газів під дією температури.
c) Фотоелектричні. Використовують напівпровідникові фотоелектричні перетворювачі. Є найефективнішим на даний час способом перетворення енергії сонця.

Процес перетворення сонячної енергії в електричну починається на фотоелектричних елементах завдяки фотоелектричному ефекту. Фотоелемент (Рис 1.1) – це два шару напівпровідників, які мають різну провідність, з боків приплавляються контакти для підключення до інших секцій або вихідної клеми. N провідник виступає катодом, а P – анодом.

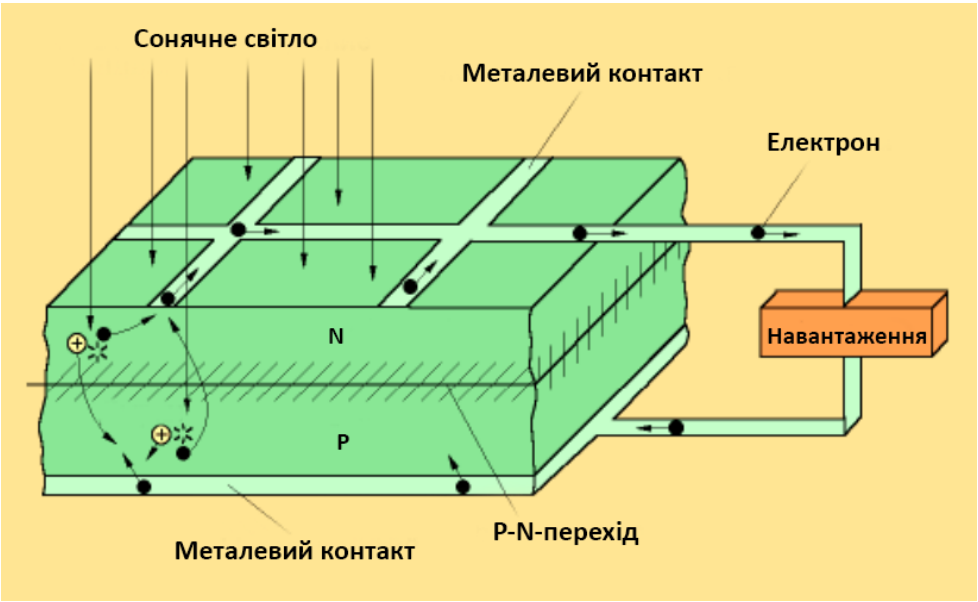


Рисунок 1.1 – Фотоелемент

Під дією сонячного світла електрони отримують додаткову енергію, яка достатня для переходу бар’єру P-N-переходу. Внаслідок чого число електронів і дірок змінюється, а отже з’являється різниця потенціалів, величина якої залежить від розміру фотоелемента.

Велика кількість фотоелементів утворюють сонячну панель, яка встановлюється на металоконструкцію, дах, трекер чи іншу жорстку поверхню. Після отримання електричної енергії, вона потрапляє на інвертор (Рис 1.2).

Рисунок 1.2 – Інвертор

Інвертор перетворює постійний струм в змінний та збільшує його амплітуду до 220 або 380 вольт. Такі моделі також слугують контролером заряду, заряджаючи акумуляторні батареї та розподіляючи енергію по споживачам.

Змінний струм після інвертора потрапляє до споживачів електричної енергії або до державної мережі через лічильник «Зеленого тарифу».

1.3 Об’єкт управління

1.3.1 Загальна характеристика об’єкта управління

Метою роботи є збільшення ефективності здобування електроенергії сонячною фотоелектричною уставкою, це дозволить зменшити витрати та збільшить щільність здобутої енергії на 1 кв.м землі. Для збільшення ефективності необхідно щоб сонячні промені падали на сонячну панель під

прями кутом якомога довше. Але це не можливо без рухомої системи, адже сонце постійно рухається і його азимут змінюється протягом дня, а висота протягом року. Конструкція яка слідкує та направляє сонячну панель, називається сонячним трекером. Вони бувають двох або одно осьові, тобто слідкування проходить або за азимутом та вистою, або тільки за азимутом. Для цієї робити був обраний одноосьовий трекер.

Схематичний вигляд об’єкта управління (Рис 1.3), представляє собою металеву, двох опорну конструкцію, яка встановлюється на поверхню землі. На кінцях опор присутні підшипники, під металеву вісь каркасу. Каркас слугує місцем кріплення чотирьох сонячних панелей, загальною потужністю 1 280 Вт.

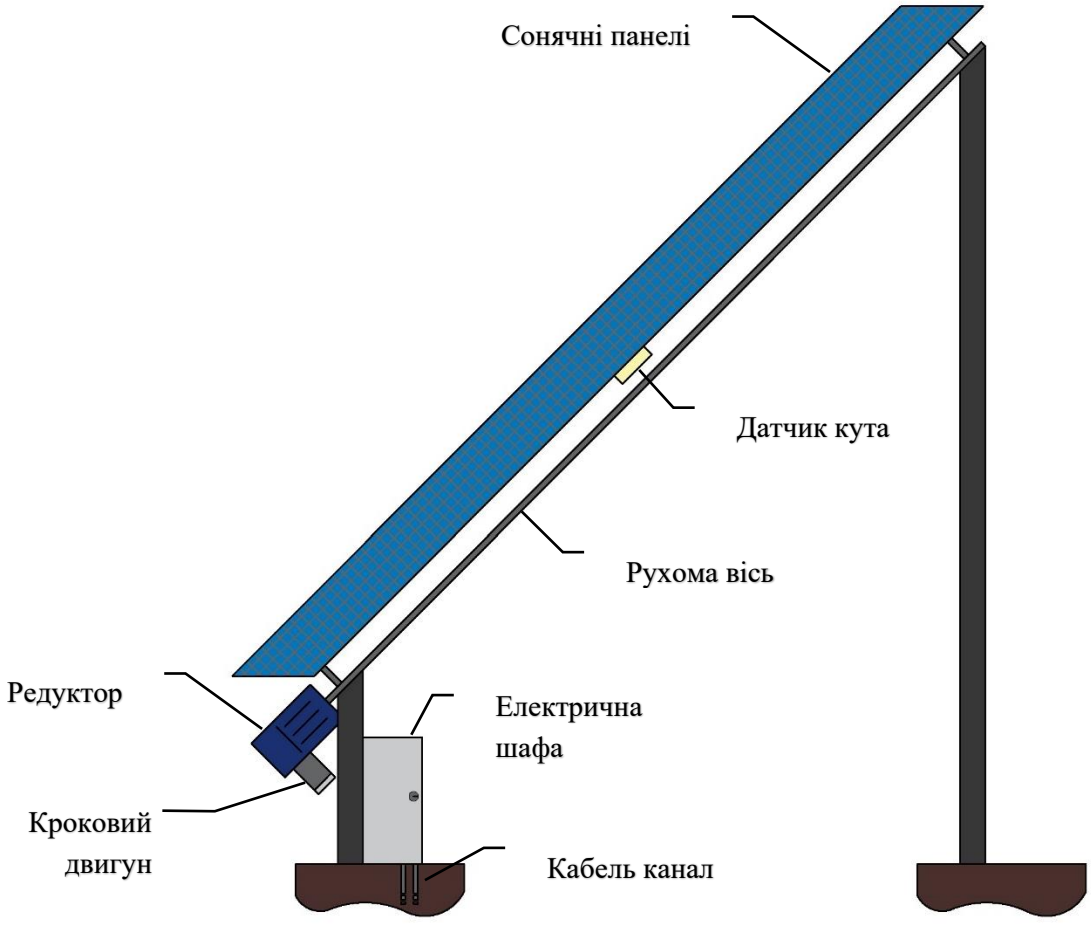


Рисунок 1.3 – Об'єкт управління

Сонячні панелі з’єднуються паралельно. Їх параметри наведені у таблиці 1.1 На каркас окрім панелей встановлюється одноосьовий датчик кута.

Таблиця 1.1 – Технічні характеристики фотоелектричних модулів

Назва параметру	**Значення**
Номінальний струм	9.7 А
Струм короткого замкнення	10.02 А
Номінальна напруга	33 В
Напруга холостого ходу	40.6 В
Вага	17.5 Кг
Розміри	1665x1002x35 мм
Технологія	Моно-кристал
Номінальна потужність	320 Вт

Каркас може вільно рухатись по одній осі, в межах від 20˚до 160˚відносно вертикальної лінії. На нижньому кінці валу встановлюється черв’яковий редуктор, до якого під’єднується кроковий двигун. На нижній балці монтується металева шафа, яка має захист не нижче ніж IP 63. В ній знаходиться плата управління трекером, датчики напруги та струму, блоки живлення та одно платний комп’ютер. Всі кабелі до установки заводяться через кабель канал.

Оскільки трекер одноосьовий, дуже важливу роль відіграє нахил металоконструкції. Для його підбору необхідно визначити зміну висоти сонця за рік, відносно координат місця знаходження конструкції. Для цього використовувались координати університету, та власноруч розроблене ПЗ. Результатом роботи є наступні данні (Рис 1.4). Вісь X – день року, Y – кут сонця. З графіку видно не тільки зміну висоти, а й протяжність світлового дня. Зелений колір день, синій ніч. С цих даних виходить, що оптимальний нахил конструкції становить 43˚- 47˚. Тому обраний нахил складає 45˚.

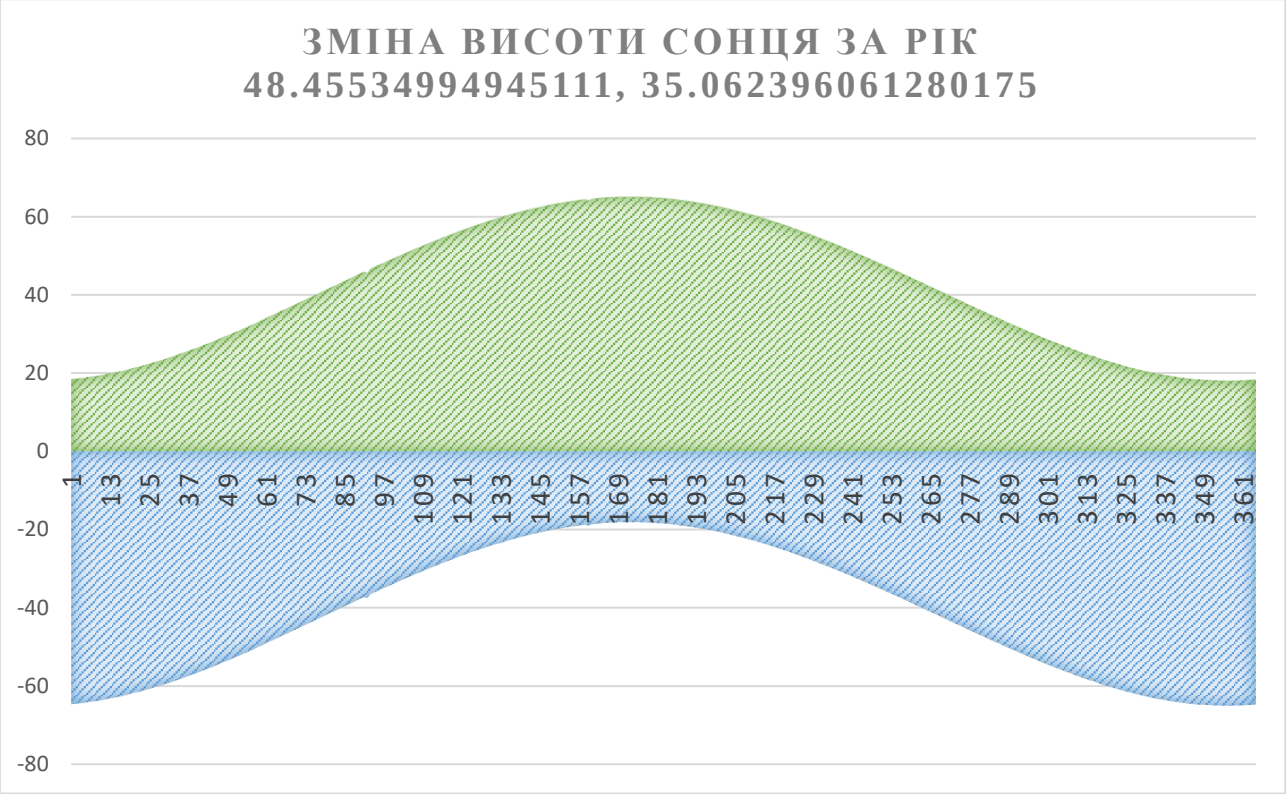


Рисунок 1.4 – Процес зміни висоти сонця

1.3.2 Структура об’єкта управління

Вхідними параметрами слугують:

- Орієнтація фотоелектричної установки.
- Напруга.
- Струм.
- Час.

Вихідні параметри:

- Напрямок руху крокового двигуна.
- Кількість кроків.

Ці параметри зчитує, первинно обробляє та створює контролюючий пристрій, на базі MCP 2210. Всі математичні розрахунки, остаточну обробку даних та створення команд управління виконує одно платний комп’ютер. Він також підключений до світової мережі, що дозволяє отримувати точний час та дистанційно керувати або слідкувати за системою. Рисунок 1.5 відображає структуру об'єкта управління.

Рисунок 1.5 – Структура об’єкта управління

1.3.3 Принцип функціонування об’єкта керування

Об’єкт управління працює у двох режимах, в залежності від того яке налаштування було обрано оператором. Ручний режим передбачає повний контроль системи людиною, оператор може самотужки обрати кут нахилу установки, після чого програмне забезпечення розраховує кількість кроків та напрямок руху для досягнення обраного кута оператором. Також в ручному режимі виводяться всі параметри системи, розраховується положення сонця але ці данні виконують лише інформативну роль.

Автоматичний режим є головним режимом роботи трекеру. Система направляє сонячні панелі за координатами сонця, автоматично знаходячи необхідний кут нахилу. У разі, якщо висота сонця менша 0˚ система перестає слідкувати та повертається на схід. Очікуючи підйома сонця. Також відбувається моніторинг вихідної потужності. Дуже мала або зовсім відсутня потужність може вказувати на хмарність, закидання снігом. В такому випадку система

спробує очистити поверхню від снігу, у разі зимового календарного місяця та близькою до нуля потужністю. Установка з максимальною швидкістю встановить найбільший кут, 20 або 160 градусів, відносно прямовисної лінії. Після чого повернеться до звичної роботи. Це дозволить очистити установку від снігу. Якщо ж цього не сталося, система відреагує як на хмарність та виставить панелі паралельно поверхності землі. І буде перебувати в такому стані до заходу сонця. Це необхідно для зменшення енерговитрат.

Методика розрахунку положення сонця полягає в знаходженні його висоти та азимуту, знаючи час та точні координати місця встановлення установки. Спочатку необхідно визначити різницю між істинним часом та середнім сонячним, тобто рівняння часу, за формулою 1.1.

$$T_{\text{рв}} = 9.87 * \sin(2B) - 7.53 * \cos(B) - 1.5 * \sin(B) \quad (1.1)$$

де $B = \frac{360° * (N - 81)}{365}$; N – номер дня року.

Потім знаходиться сонячний час, формула 1.2 та часовий кут світила 1.3. T_{UTC} – це реальний час у форматі UTC, α – довгота.

$$T_{\text{ic}} = T_{\text{рв}} + T_{UTC} + \alpha \quad (1.2)$$

$$t = T_{\text{ic}} - 720 \quad (1.3)$$

Знаючи ці дані та зміну нахилу сонця відносно екватору можемо знайти висоту та азимут. Висота – це кут між лінією направленою на сонце та математичним горизонтом. А кут між лінією перетину математичного горизонту з площиною вертикального круга світила та прямовисною лінією називається азимутом (Рис 1.6). Формули їх знаходження представленні нижче, для азимуту 1.4, для висоти 1.5.

$$A = atan2\,(\cos(\delta) * \sin(t),\ \cos(\delta) * \sin(\varphi) * \cos(t) - \sin(\delta) * \cos(\varphi)) \quad (1.4)$$

$$H = \text{asin}(\sin(\delta) * \sin(\varphi) + \cos(\delta) * \cos(\varphi) * \cos(t)) \quad (1.5)$$

Де φ – широта; t – часовий кут; δ – нахил сонця.

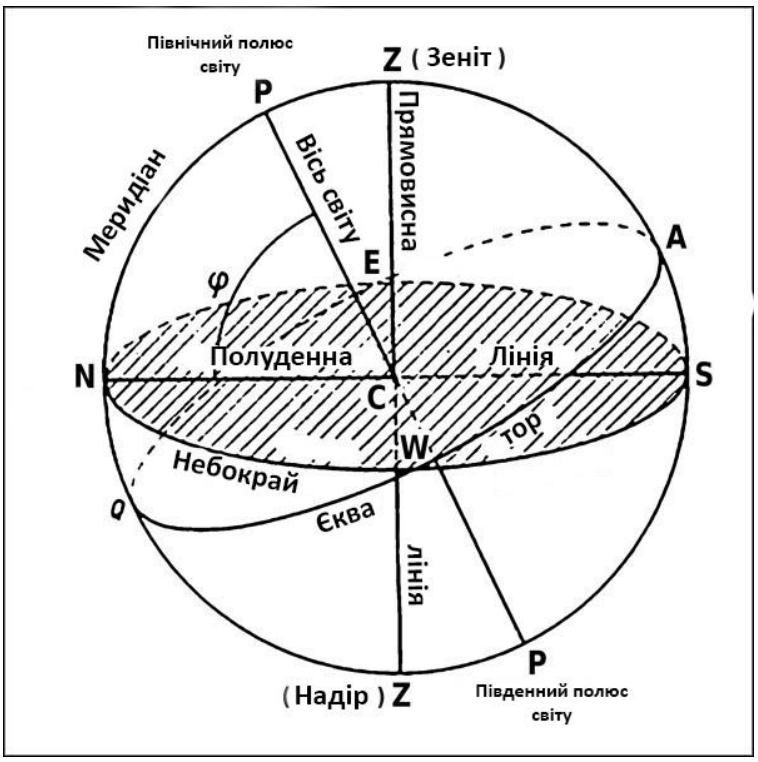


Рисунок 1.6 – Небесна сфера

1.4 Формування мети розробки

Основною задачею є розробка системи управління, підбір давачів та виконавчих пристроїв, розробка принципової схеми. Створення алгоритму роботи системи та написання програмного забезпечення. З метою збільшення ефективності здобутку енергії та зменшення економічних витрат.

1.5 Висновки за розділом

Представлена система є дискретною. А отже потребує розробки програмного комплексу для виконання заданих у розділі функцій.

Основними критеріями розробки ПЗ є:

- Інформативність. Програмне забезпечення повинно мати зрозумілий інтерфейс та правильно описанні функції.
- Велика точні розрахунків. Необхідно забезпечити рівень позиціювання з точністю 1˚.
- Збирання та обробку даних. Забезпечуючи можливість аналізу ефективності системи.

- Віддалене управління. Надати можливість відстежувати та керувати системою на відстані.

Окрім цього система повинна мати високу надійність та відмово стійкість. Необхідно вірно підібрати виконавчий пристрій за механічними критеріями конструкції. Забезпечити апаратні системи захисту від втрат зв’язку з комп’ютером. Та розрахувати електронні елементи які відповідають технічним особливостям об’єкта керування. Опиратися на оптимальний рівень економічних витрат. Виявити та прийняти рішення щодо запобігання травмування на робочому місці, та прикласти усі можливі зусилля для збільшення робочого комфорту.

2 РОЗРОБКА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ УПРАВЛІННЯ

2.1 Розробка структурної схеми системи управління

Для розробки структурної схеми необхідно визначити вхідні та вихідні параметри. Оскільки об’єктом управління є трекер, для зміни нахилу використовується кроковий двигун. Тоді вихідними параметрами виступає напруга, яка подається імпульсами на обмотки двигуна, а частота та порядок подання цих імпульсів визначають напрямок та швидкість руху. Для спрощення схеми та підвищення якості керування використовується драйвер крокового двигуна TB 6600, який знаходиться на пристрої управління. Вихідними параметрами виступає кут нахилу сонячних панелей від аналогового інклінометру, та потужність яка формується з двох параметрів, струму від датчику на ефекті Холла та напруги. Відповідно до цього наведена структурна схема 2.1.

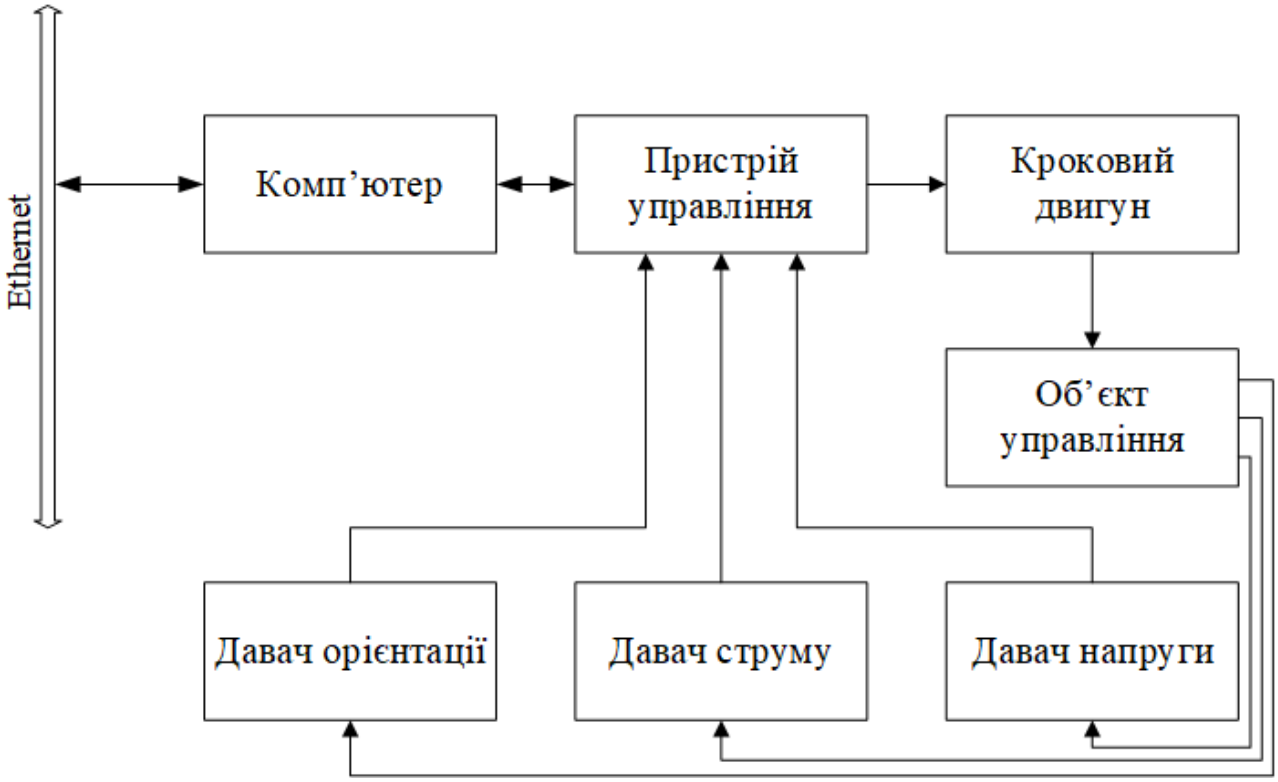


Рисунок 2.1 – Структурна схема системи управління

2.2 Розробка структурної схеми інформаційних потоків

Система окрім вирішення задачі автоматизації повинна мати додаткові функції, які спрямовані на підвищення якості та оптимізації виробництва енергії. Підвищення робочого комфорту, та збільшення конкурентного впливу. Для цього необхідно:

- Забезпечити збір інформації з датчиків відповідно до інерційності системи.
- Створити системи контролю обладнання.
- Організувати протиаварійні заходи.
- Архівувати параметри установки.
- Візуалізувати людино-машинний інтерфейс.
- Реалізувати ручний режим роботи.

Окрім цього, система повинна вміти працювати з мережевими ресурсами, мати віддалене управління, керувати обладнанням. Для реалізації функцій використовується середовище розробки Visual Studio 2022 та мова програмування C#. Все це формує програмний комплекс для робити на комп'ютері з операційною системою Windows 7 або вище.

Зв’язок з апаратним комплексом забезпечується за допомогою загально промислової мережі Ethernet, а обмін між модулями по Universal Serial Bus (USB), Serial Peripheral Interface (SPI). В програмному забезпеченні передача відбувається за допомогою стандартних протоколів.

Представлена структурна схема (Рис 2.2) забезпечує позиціонування сонячних елементів, збір та зберігання їхніх параметрів, захист від аварійних ситуацій й можливість дистанційного керування.

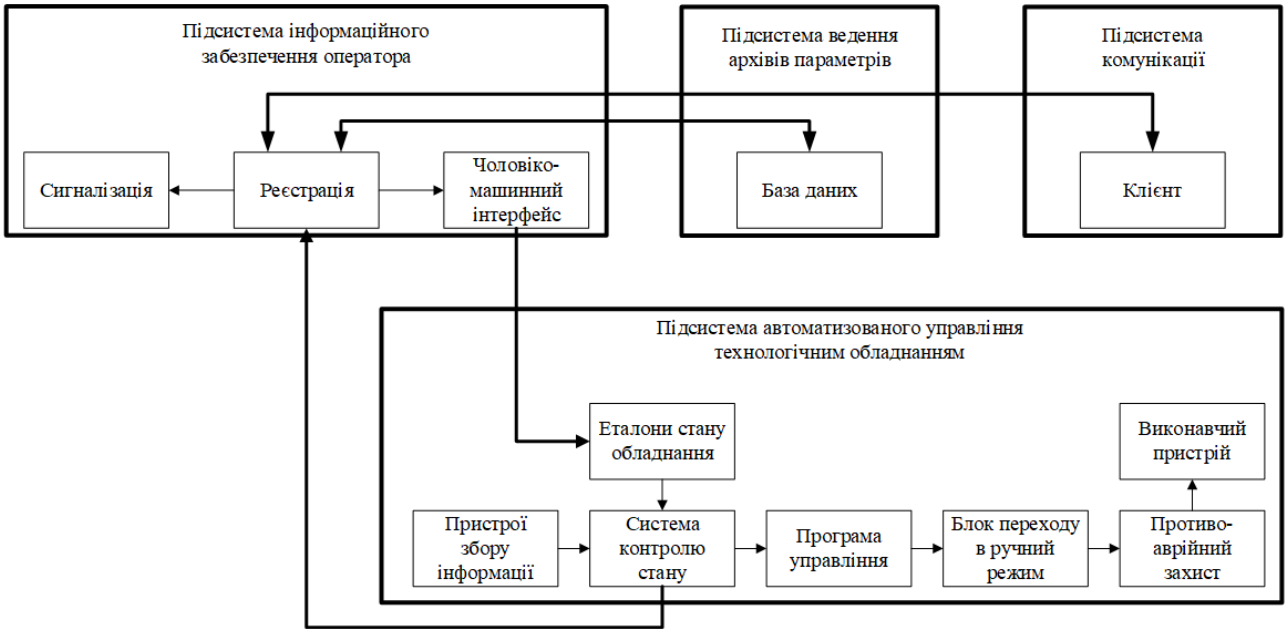


Рисунок 2.2 – Структурна схема інформаційних потоків

2.3 Вибір апаратного забезпечення системи управління

2.3.1 Вибір давачів

Для позиціонування та для організації зворотного зв’язку необхідно знати кут нахилу сонячних елементів. Установка може змінювати свій кут нахилу лише в одній площині та в межах від 20 до 160 градусів відносно вертикальної лінії. Для отримання цього параметру використовується аналоговий інклінометр IN81 (Рис 2.3). Оскільки цей інклінометр має багато варіантів, був обраний одноосьовий з вихідним сигналом у вигляді зміни напруги. За допомогою датчика, можливо міряти кути обертання по одній осі від 0 до 180 градусів. Та має налаштування параметрів, таких як встановлення 0, зміни діапазону вихідного сигналу. Характеристики датчику представлені таблицею 2.1.

Interface	Type of connection	M12 connector, 8-pin									
2, 3, 4, 5 voltage	1	Signal:	0 V	+V	Uout+	Uout-	$\overline{\text{Uout+}}$	$\overline{\text{Uout-}}$	Teach 1	Teach 2	
		Pin:	1	2	3	4	5	6	7	8	

Рисунок 2.3 – Інклінометр Kubler IN 81

Таблиця 2.1 – Технічні характеристики інклінометру

Назва характеристики	**Значення**
Напруга живлення	10-30 VDC
Споживаючий струм	30 mA
Робоча температура	-40…+85 ℃
Матеріал	Алюміній
Тип захисту	IP 65
Тип вихідного сигналу	Аналоговий
Діапазон зміни вихідного сигналу	0 – 5 V
Діапазон вимірювання	0˚- 180˚(0˚- 360˚)

Окрім визначення рівня нахилу, для організації функції збору даних та деяких режимів роботи установки, необхідно знати потужність здобування енергії. Для цього необхідно вимірювати струм та напругу. Оскільки сонячні

панелі з’єднуються паралельно то максимальна напруга сягає 40.6В, а струм 40.08А. Для виміру струму використовується датчик ACS 714-50A-T (Рис 2.3) виробництва Allegro. Він здатний виміряти струм до 50А, та працює на ефекті Холла. Вихідний сингал змінюється від 0.5 до 4.5В, відповідно до діапазону виміру -50…+50А. 2.5В це 0А. Детальні параметри у таблиці 2.2.

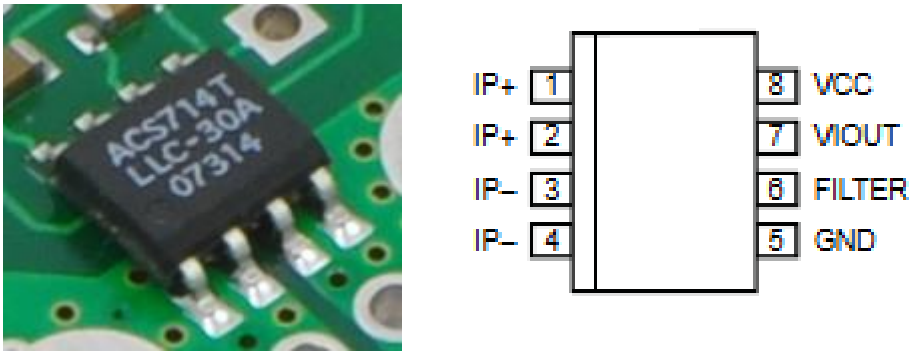


Рисунок 2.3 – Датчик струму ACS 714-50A-T

Таблиця 2.2 – Технічні характеристики датчику струму

Назва характеристики	**Значення**
Напруга живлення	5 VDC
Споживаючий струм	10 mA
Робоча температура	-40…+150 ℃
Корпус	SOIC-8
Матеріал	Пластик
Тип вихідного сигналу	Аналоговий
Діапазон зміни вихідного сигналу	0.5 – 4.5 V
Діапазон вимірювання	-50…+50A

Вимір напруги здійснюється завдяки АЦП MCP 3302 (Рис 2.4), до якого встановлюється дільник напруги та операційний підсилювач LM358 в якості повторювача. Діапазон виміру залежить від опорного джерела живлення в якосьті такого виступає MCP 1525 (Рис 2.5), на 2.5В тому діапазон вимірювання АЦП сягає 0…2.5В. Їх характеристики додатково наведенні у таблицях 2.3 і 2.4 відповідно.

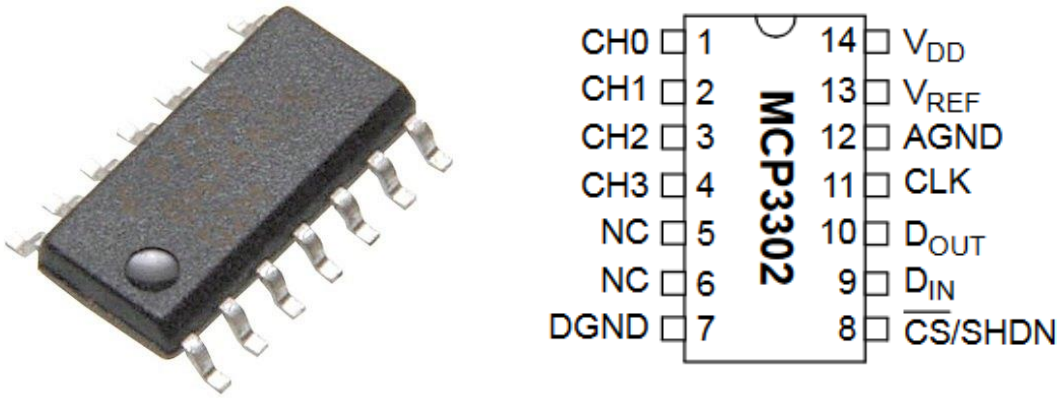


Рисунок 2.4 – АЦП MCP 3302

Таблиця 2.3 – Технічні характеристики АЦП

Назва характеристики	**Значення**
Напруга живлення	4.5…5.5 VDC
Споживаючий струм	4.5 mA
Робоча температура	-40…+85 ℃
Корпус	SOIC-14
Матеріал	Пластик
Тип вихідного сигналу	Цифровий (SPI)
Діапазон вимірювання	0…2.5В
Розряд АЦП	13-bit
Кількість каналів	4

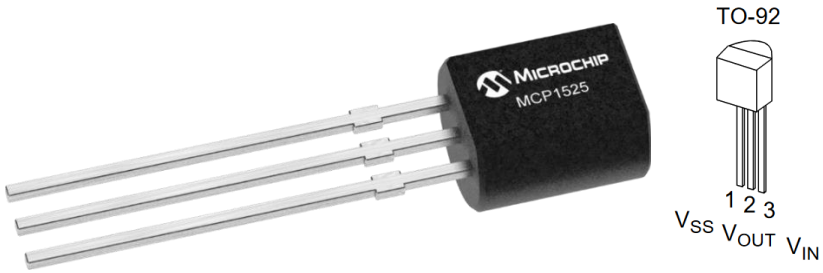


Рисунок 2.5 – Джерело опорної напруги MCP 1525

Таблиця 2.3 – Технічні характеристики джерела опорної напруги

Назва характеристики	**Значення**
Напруга живлення	2.7…5.5 VDC
Споживаючий струм	100 uA
Робоча температура	-40…+85 ℃
Корпус	TO-92
Матеріал	Пластик
Вихідна напруга	2.5В

Загальна кількість датчиків та їх основні характеристика наведені у таблиці 2.4.

Таблиця 2.4 – Загальні параметри датчиків

Назва параметру	Тип	Діапазон змінення	Точність	Період оновлення	Напруга живлення	Потужність споживання
Кут нахилу	Аналоговий	20˚- 160˚	0.044˚	5 Гц	24В	0.72 Вт
Напруга панелей	Цифровий	0-40.6В	0.01В	5 Гц	5В	0.024 Вт
Струм панелей	Аналоговий	0-50А	0.012А	5 Гц	5В	0.05 Вт

2.3.2 Вибір виконавчих пристроїв

Для зміни нахилу використовується кроковий двигун, з черв’яковим редуктором. Щоб підібрати вірно редуктор та двигун необхідно дізнатися момент, який необхідний для повороту. Розмір каркасу з сонячними панелями становить 3350х2020 міліметрів, а його вага близько 80 кг.

$$I = \frac{m * l^2}{12} = \frac{80 * 2.02^2}{12} = 27.2\ H/\text{м}$$

В якості крокового двигуна використовується Nema 34 6Nm (Рис 2.6). Який має 6 $H/$м обертального моменту, але після редуктора(Рис 2.7) 10:1 ми

отримаємо 60 $H/$м. А розмір кроку становитиме 0.18˚. Параметри, таблиці 2.5 та 2.6.

Рисунок 2.6 – Крокoвий двигун Nema 34

Таблиця 2.5 – Характеристики крокового двигуна

Назва характеристики	**Значення**
Обертаючий момент	60 кг/см (6 Н/м)
Споживаючий струм	4 А
Кут кроку	1.8˚
Кількість фаз	2
Матеріал	Сталь
Індуктивність	4 mH

Для збільшення моменту та блокування конструкції від мимовільного повороту застосовується черв’яковий редуктор.

Рисунок 2.7 – Черв’яковий редуктор RV40-86

Таблиця 2.6 – Характеристика редуктора

Назва характеристики	**Значення**
Співвідношення	10:1
Кількість ступенів	1
Обертаючий момент	90 Н/м
ККД	77%
Вхідне кріплення	86 мм (NEMA 34)

Для керування кроковим двигуном використовується драйвер TB6600HG (Рис 2.8), який розрахований на струм до 5А та має зручну систему управління. Він забезпечує комутацію двигуна та його реверс, завдяки зміні порядку подання та частоти імпульсів.

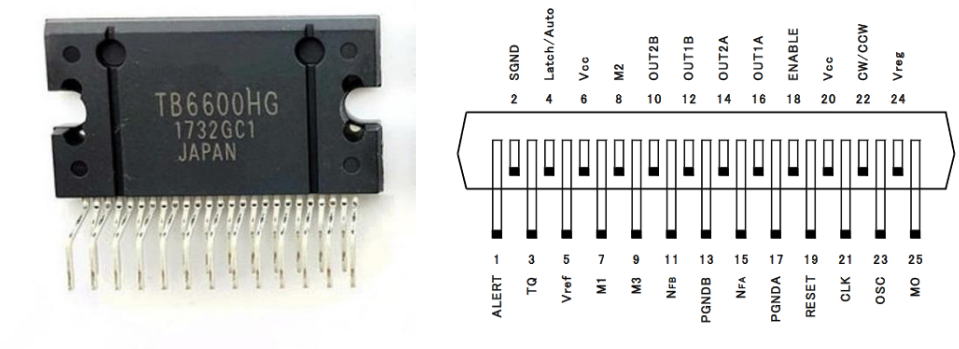


Рисунок 2.8 – Драйвер крокового двигуна TOSHIBA TB6600HG

Таблиця 2.7 – Параметри драйверу двигуна

Назва характеристики	**Значення**
Напруга живлення	12 – 50В
Максимальний струм	5А
Корпус	HZIP25-P-1.00F

2.3.3 Вибір пристроїв управління

Система є досить інерційною та не потребує швидкого реагування на події, але пристрій управління має відповідати мінімальним системним потребам

програмного забезпечення, та мати необхідні апаратні рішення. Тому в якості пристрою управління виступає комп’ютер Z83II (Рис 2.9). Його характеристики наведені у таблиці 2.8. Він повністю відповідає апаратним потребам, а саме наявність USB 2.0 та мережевої карти с роз’ємом RJ-45. Для встановлення ПЗ й коректної роботи, необхідно мати операційну систему не нижче Windows 7, 1 Гб вільного місця, 512 мб оперативної пам’яті, процесор не нижче ніж Intel Pentium 4 610 або AMD Athlon 64 3200+.

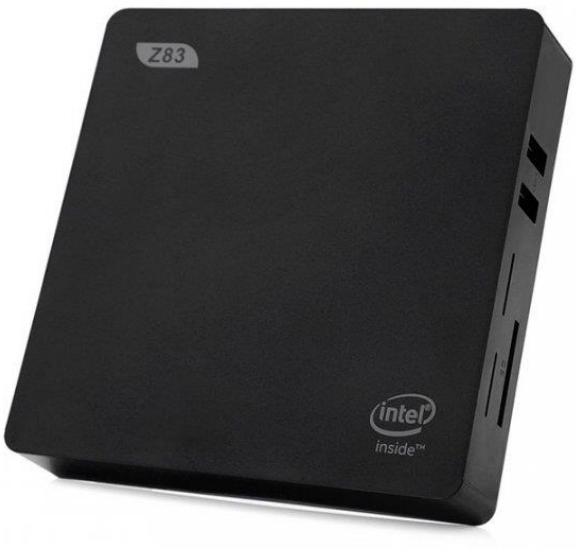


Рисунок 2.9 – Одноплатний комп’ютер Beelink Z83

Таблиця 2.8 – Характеристики комп’ютера

Назва характеристики	**Значення**
Напруга живлення	12 В
Струм живлення	2А
Процесор	Intel Atom x5-Z8350
Кількість ядер	4
Максимальна частота	1.9 ГГц
Обсяг оперативної пам’яті	2 Гб
Обсяг постійної пам’яті	32 Гб
Операційна система	Windows 10 LTSC

Продовження таблиці 2.8

Назва характеристики	Значення
Роз’єми	- 1 x HDMI - 1 x USB 3.0 - 3 x USB 2.0 - 1 x RJ-45 - 1 x аудіо порт 3.5мм - 1 x micro – USB 1 x microSD

Комп’ютер використовується для розрахунків, аналізу та управляння параметрами, реалізації збереження даних та людино-машинного інтерфейсу. Але для керування установкою потрібно додати додатковий модуль який розроблений власноруч на базі MCP 2210 (Рис 2.10).

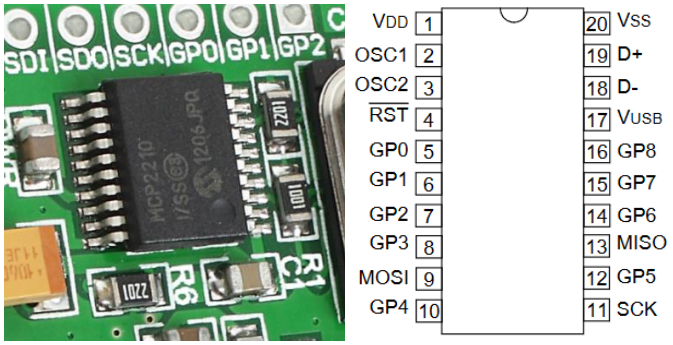


Рисунок 2.10 – USB – SPI міст MCP 2210

Таблиця 2.9 – Характеристика модуля MCP 2210

Назва характеристики	Значення
Напруга живлення	3.3 – 5.5 В
Корпус	SSOP - 20
Версія USB	USB 2.0 Full Speed
Сумісність з операційними системами	Windows XP or later, Linux, Mac OS
Кількість GP	9
Обсяг EEPROM	256 байт
Кількість SPI	1
Робоча температура	-40…+85 ℃

Модуль підключається до комп’ютера по роз’єму USB(Рис 2.11) та зчитує данні з терекеру і забезпечує керування кроковим двигуном. Схема цього модулю зображена на рисунку 2.12.

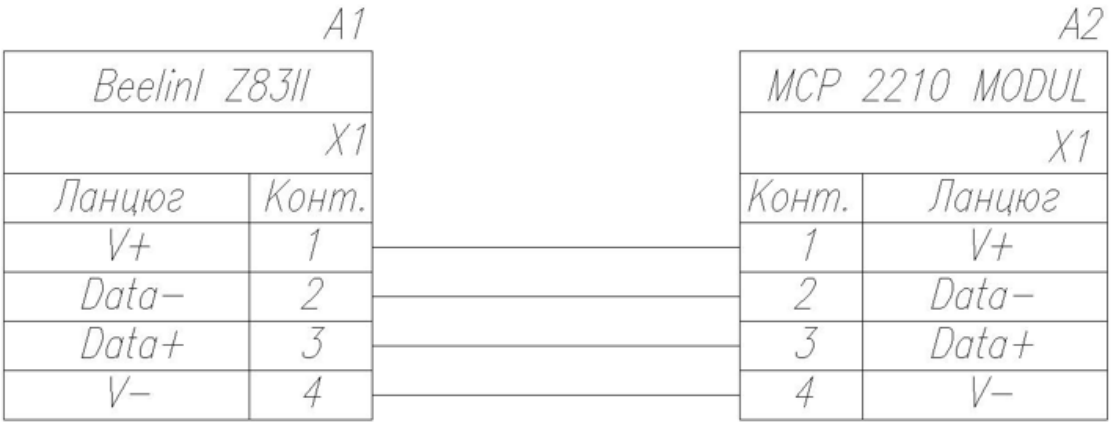


Рисунок 2.11 – Схема підключення

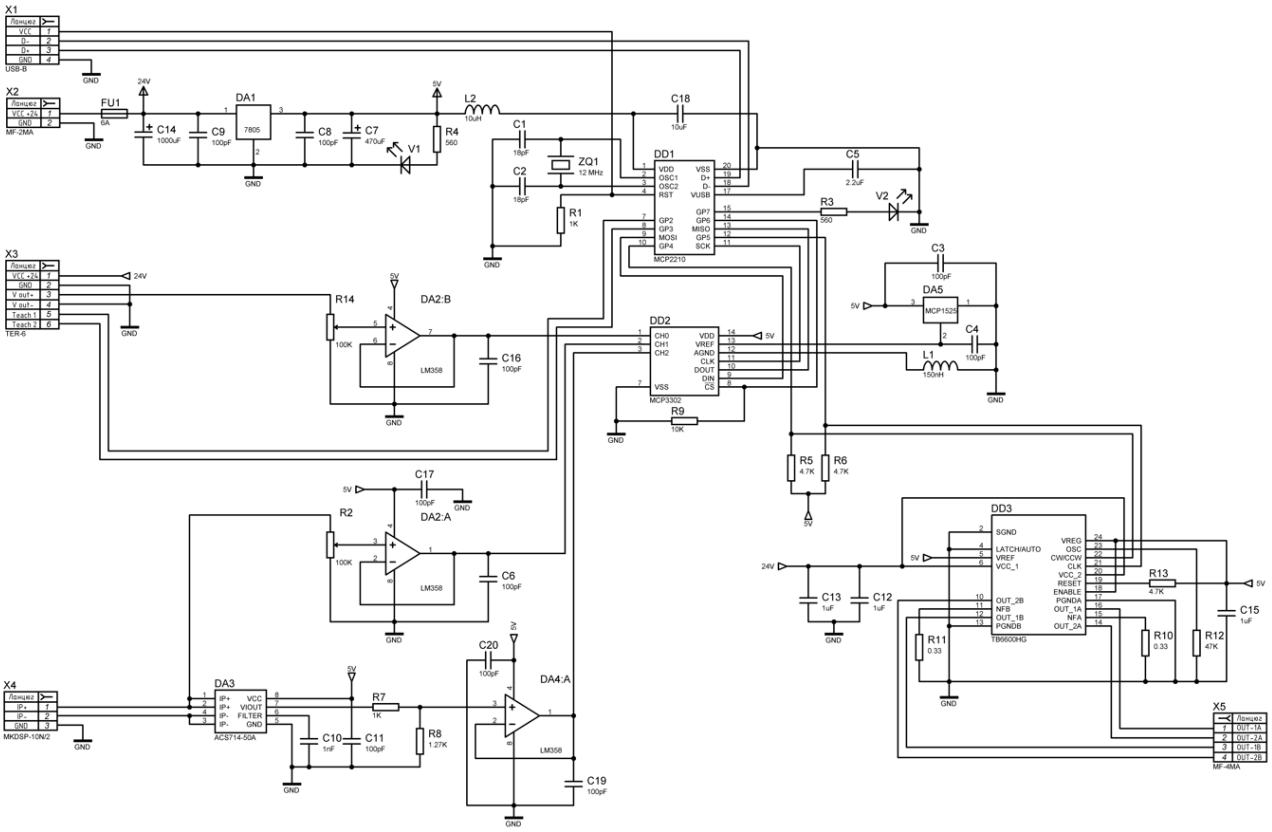


Рисунок 2.12 – Схема модуля

На модулі знаходиться вся необхідна обв’язка для MCP 2210, фільтр для лінії зв’язку, кварцовий резонатор, ланцюг скидання. Який необхідний для встановлення усіх портів в логічний 0, при відсутності підключеного USB

кабелю. Всі споживачі на 5В живляться з DA1, це лінійний стабілізатор напруги. Розрахований на 1.5А, усі прилади які підключені до лінії не споживають більше 100 mA.

Вимір усіх параметрів відбувається завдяки MCP 3302, DD2. Цей 13 бітний АЦП має зовнішнє джерело опорної напруги DA5 на 2.5В. Другий канал АЦП зчитує напругу сонячної панелі, яка поступає на клему X4 проходить через змінний резистор R2. Який представляє собою змінний дільник напруги. Після чого зменшена напруга з сонячної панелі потрапляє на операційний підсилювач, який увімкнутий як не інвертуючи повторювач, це необхідно для узгодження низько омного входу АЦП та високо омного виходу дільника. З виходу ОП сигнал потрапляє до АЦП де відцифровується та передається по лінії SPI до MCP 2210. Вимір кута нахилу відбувається датчиком IN81, який підключається до клемника X3. Датчик живиться з модулю, та має вихідний сигнал у вигляді зміни напруги 0 – 5В. Схема виміру напруги аналогічна до попередньої, її елементи мають однакове призначення. Вимір струму проходить завдяки мікросхемі DA3. ACS 714 – 50A повертає виміряну величину у вигляді напруги 0.5 – 4.5В, для виміру необхідного діапазонну встановлений дільник напруги с коефіцієнтом 1.79. Що забезпечує при вхідній напрузі у 4.5В видати 2.51В.

Керування кроковим двигуном відбувається завдяки драйверу TB 6600, це спрощує схему управління та дозволяє задіяти лише два виходи MCP 2210. Один вихід підключається до CW/CCW, що забезпечує заміну напрямку руху. 0 – вліво, 1 – вправо. Другий вихід підключається до CLK, один імпульс на контакті дорівнює одному кроку двигуна. У нашому випадку це 1.8 градуса. Інші функції драйвера не використовуються.

Після обрання модулів та пристрою управління була складена загальна таблиця споживання потужності 2.10.

Таблиця 2.10 – Загальне споживання

№	Назва модулю	Пристрій	Напруга живлення	Потужність споживання
1	Beelink Z83	Комп’ютер	12 В	24 Вт
2	MCP 2210	Модуль управління	5 В	0.4 Вт
		Датчик нахилу IN81	24 В	0.72 Вт
		Датчик струму ACS 714	5 В	0.024 Вт
		Датчик напруги MCP 3302	5 В	0.05 Вт
		Кроковий двигун NEMA 34 6Nm	24 В	100 Вт

2.3.4 Вибір пульта оператора

В якості пульта оператора може виступати любий комп’ютер. Встановлене програмне забезпечення на пристрій управління дозволяє реалізувати всі необхідні функції, які були зазначені в першому розділі та в схемі інформаційних потоків. Інший комп’ютер може підключитися до віддаленого робочого столу пристрою управління. Або можна використовувати мобільний додаток, якщо нема необхідності у повному списку функцій.

2.3.5 Вибір джерела живлення

В комплекті з одноплатним комп’ютером поставляється блок живлення (Рис 2.12) на 12В 2А. Тому доцільніше використовувати саме його.

Рисунок 2.13 – Комплектний блок живлення

Для модуля розширення необхідно підібрати блок відповідно до потужності що споживається. Оскільки 5В отримується завдяки лінійному стабілізатору напруги з 24В, то потужність всіх 5 В приладів на вході стабілізатору становить 2.275 Вт.

$$P_{\text{БЖПК}} = 1{,}3 \cdot (100 + 0.72 + 2.275) = 134 \text{ Вт}$$

Обраний блок живлення Mean Well 156W DC24V (Рис 2.13), а його характеристики наведенні у таблиці 2.11.

Рисунок 2.14 – Блок живлення LRS-150-24

Таблиця 2.11 – Параметри блока живлення

Назва характеристики	**Значення**
Вхідна напруга	170 – 264 VAC
Вихідна напруга	24 В
Потужність	156 Вт
Рівень захисту	IP 20
Присутність захисту	- Від перегріву - Від перенавантаження - Від перенапруги - Від короткого замкнення
Робоча температура	-40…+70 ℃

2.4 Розробка функціональної схеми автоматизації

На підставі вижче описаного була створена схема автоматизації (Рис 2.15).

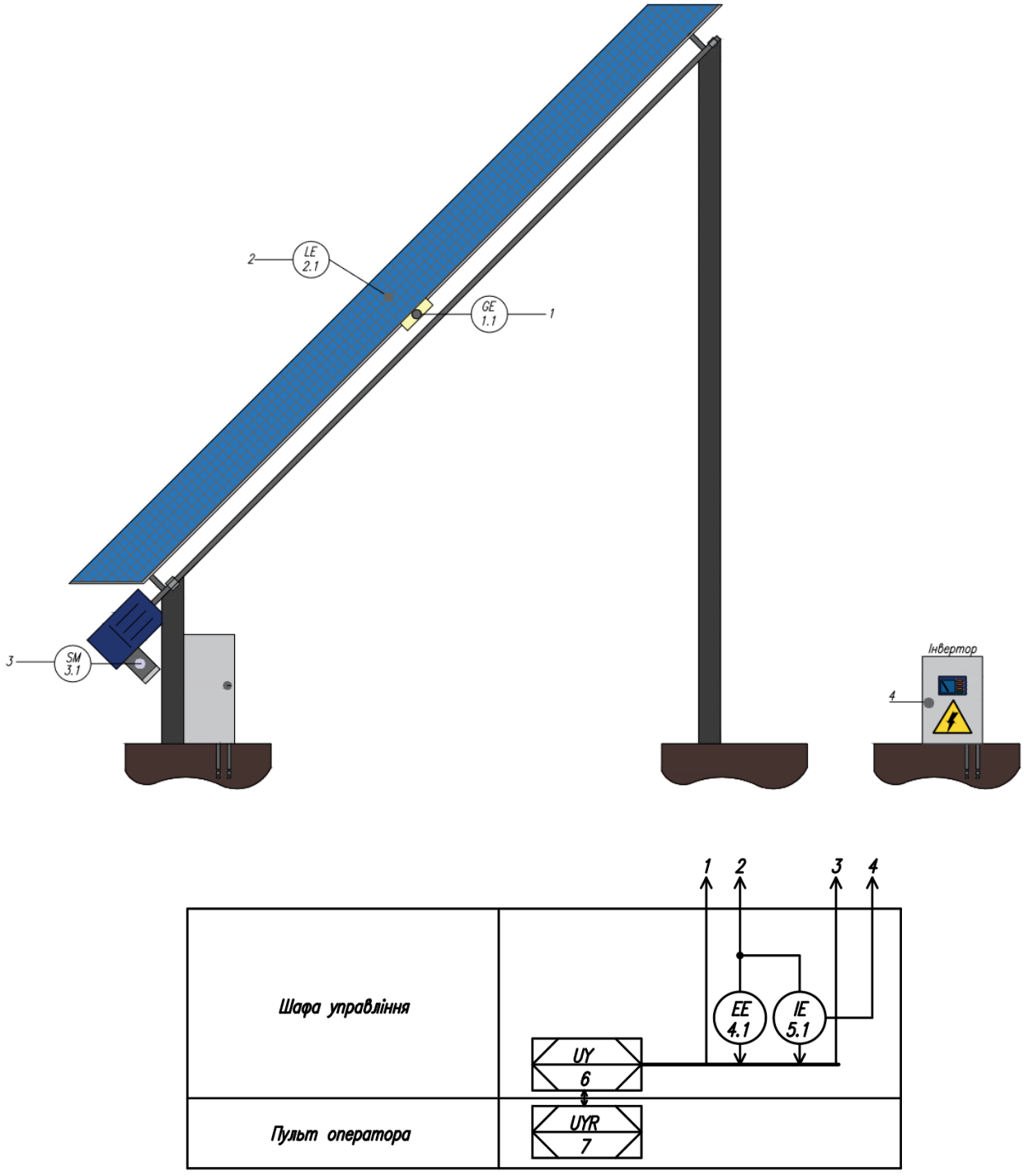


Рисунок 2.15 – Функціональна схема автоматизації

Управління здійснюється одноплатним комп’ютером та його модулем (UY). Данні про протікання технологічного процесу можуть відображатися на

пульті оператора, в якості якого може виступати любий персональний комп’ютер (UYR) підключений до мережі інтернет. Система управління керує кроковим двигуном (SM 3.1 – NEMA 34), позиціонуючи сонячну панель відповідно до координат сонця. Інклінометр (GE 1.1 – IN81) слугує для отримання орієнтації установки. Це необхідно для зворотного зв’язку, та реалізації алгоритмів захисту, збільшення точності роботи. Датчик (EE 4.1 – MCP 3302) та (IE 5.1 – ACS714) використовуються для непрямого виміру потужності з фотоелектричних модулей (LE 2.1), яка необхідна для реалізації додаткових функцій, що здатні знизити економічні витрати. З датчика струму(IE 5.1) також йде силова електрична лінія до інвертору (4), який перетворює напругу та відає її у мережу.

2.5 Розробка схеми електричних з’єднань

Враховуючи всі давачі, виконавчі пристрої, розроблені модулі, та функціональну й принципову схеми була розроблена схема електричних з’єднань (Рис 2.16). В системі використовується два блоки живлення G1 MW LRS-150-24 та G2 DC 12-24W, вони під’єднуються до мережі 220В та забезпечують необхідними напругами всю систему. G1 підключено до модулю управління MCP 2210 (A2 - X7.2). G2 підключений до однопалатного комп’ютера Beelink Z83II (A1 – X3.3). Передача даних між комп’ютером та модулем керування реалізована через інтерфейс USB (X3.1 – X7.1).

До модуля управляння підключено єдиний виносний датчик кута (X4.1 – X7.3) та виконавчий пристрій, яким є кроковий двигун (X7.5 – M1). Земля сонячних модулів під’єднується до загальної землі схеми, а плюсовий контакт підключається до клеми вимірювача струму та напруги, який знаходиться на модулі, через клему (X5.1 – X7.4), після чого потравляє на вхідну клему інвертора AXIOMA ISPWM 2000 (X7.4 - X6.2), який підвищить до 220В змінної напруги, та від дасть її високовольтній мережі.

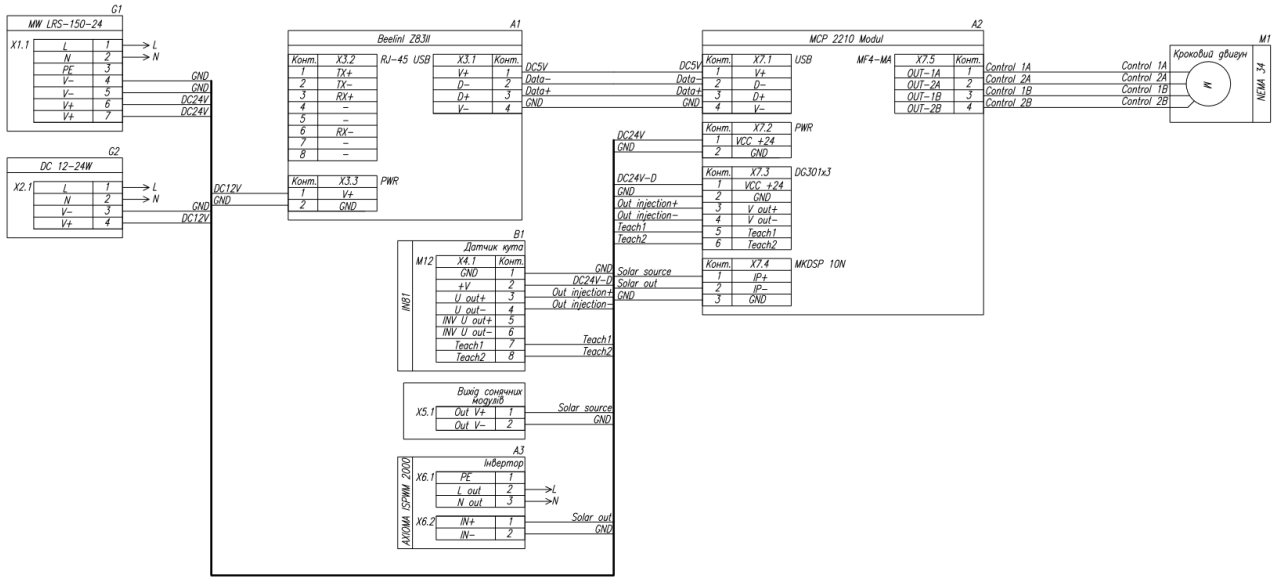


Рисунок 2.16 – Електрична схема з’єднань

2.6 Висновки за розділом

Для нашого об’єкта управління було підібране все необхідне обладнання наведені його характеристики та опис схем. А саме розроблена принципіальна схема модуля, функціональна схема автоматизації, схема електричних з’єднань.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка алгоритму функціонування системи керування

Відповідно до опису системи був розроблений алгоритм роботи. Після подачі живлення та запуску пристрою управління, встановлене програмне забезпечення отримує повний системний час у форматі UTC та починає розраховувати положення сонця відносно координат які вписані в програму у вигляді констант. Паралельно з цим система переходить у режим який був обраний оператором, ручний або автоматичний, та визначає кут установки. У разі обрання ручного керування система очікує зміни режиму роботи або подію зміни нахилу оператором, через людино-машинний інтерфейс. Якщо кут нахилу було змінено то система розраховує необхідний напрямок та кількість кроків, які потрібні для встановлення обраного кута. Й починає рухати установку

відповідно параметрам. Після завершення руху система повертається до очікування нових команд від людини.

В тому випадку коли система знаходиться в автоматичному режимі роботи, вона сама позиціонує установку відповідно до азимута сонця, цей процес є циклічним. Після встановлення кута система очікує зміни азимуту на 1 градус, що відбувається приблизно кожні 284 секунди. Цей процес завершиться у тому випадку, якщо оператор змінить режим роботи або висота сонця стане меншою за 0. Що буде свідчити про настання темряви, в цій ситуації установка перейде в початкове положення та буде очікувати сходу сонця, після чого процес повторюється.

В системі реалізовані додаткові функції, які доступні в автоматичному режимі, це очищення від снігу та перевірка на наявність хмарності. Слідкування за хмарністю відбувається за потужності та даними про погоду від метеостанції, які отримуються через інтернет. Якщо установка перебуває у нерухомому стані, а її потужність дуже мала, при цьому хмарність присутня, установка займає середнє положення відносно крайніх точок, тобто перестає слідкувати за сонцем, до тих пір поки не зникне хмарність або не зайде сонце. Алгоритм очищення від снігу схожий на хмарність, система отримує данні про низьку або відсутню потужність та інформацію про снігопади за останні 24 години. Якщо ці умови виконуються система встановлює максимально крутий кут нахилу, та повертається до нормальної роботи. У разі якщо це не допомогло збільшити ефективність система реагує також як на хмарність. Для цих двох режимів є одна спільна умова, це висота сонця більше або дорівнює 0.

За описаною вище інформацією, був створений граф станів (Рис 3.1), який повністю відповідає моделі дискретного об’єкта, скінченного автомату. Граф має наступні стани: S0 – Ініціалізацію усіх параметрів та початок розрахунку координат. S1 – Ручний режим роботи установки. S2 – Стан зміни нахилу. S3 – Автоматичний режим роботи. S4 – Перехід у базове положення. S5 – Скидання снігу. S6 – Стан під час хмарності. Вихідною змінною є SM –сигнал руху двигуна. А вхідними: WM – режим роботи, TS – подія зміни кута, WT – наявність

робочого часу, PL – низька потужність, CL – хмарність (за погодою), IS – наявність снігу (за погодою).

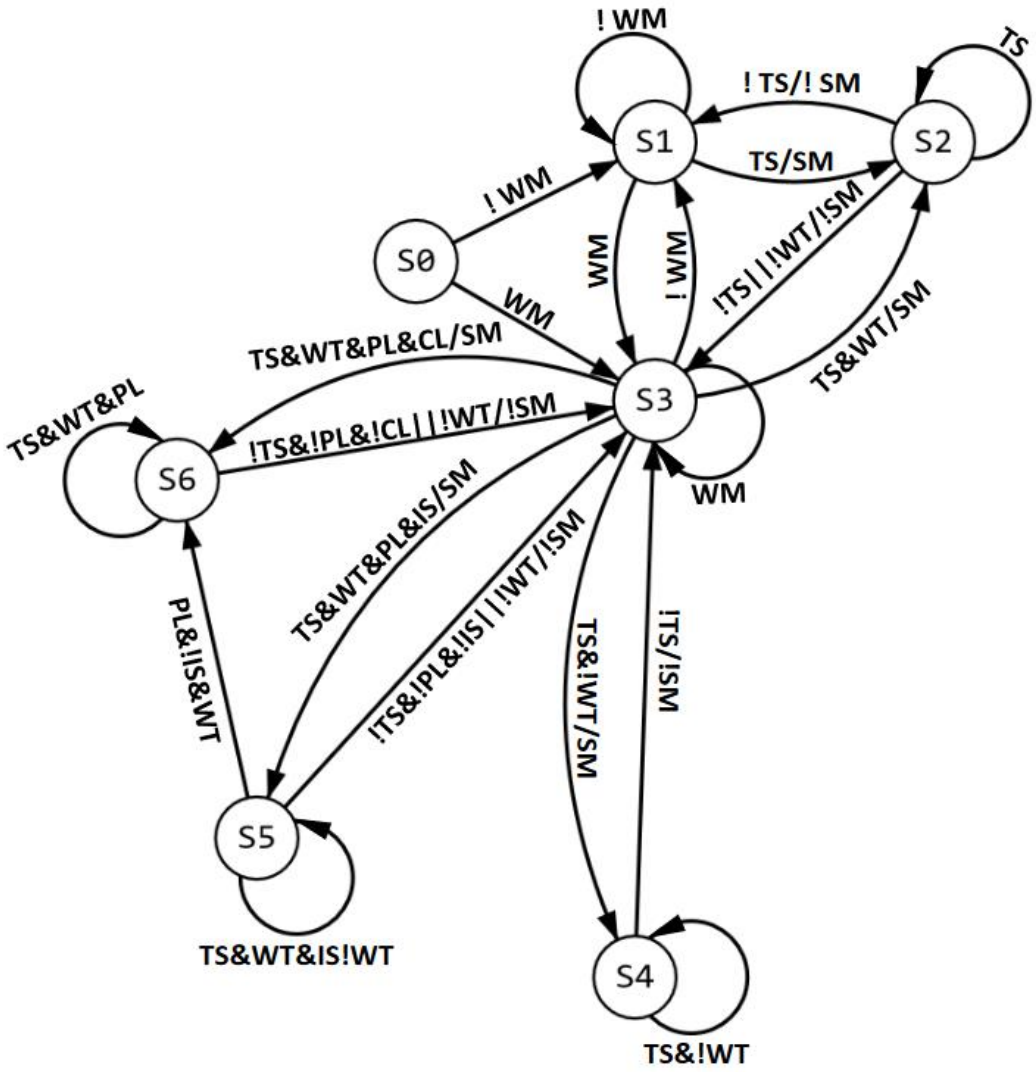


Рисунок 3.1 – Граф станів об’єкту керування

3.2 Конфігурування пристрою керування

Розроблене програмне забезпечення не потребує конфігурування однопалатного комп’ютера та буде безпомилково виконувати свою роботу у разі відповідності мінімальних системних вимог. Нижче наведений код підключення до модуля MCP 2210 на мові програмування C#, це є повний код функції UsbConnect() яка входить до класу HardwareMCP2210. Детальніше з яким можна ознайомитися в додатку. Данна функція побудована з використанням офіційної

бібліотеки Microchip для роботи з модулем. Опис основних функцій які входять до цієї бібліотеки наведено в таблиці 3.1.

```
int connect_error_ST = MCP2210.M_Mcp2210_GetConnectedDevCount(base_vid, PID);
    if (connect_error_ST == 0)
    {
      return "Module not found";
    }
    else if (pid_modul.Contains(PID))
    {
            return "Module already connected";
    }
    else
    {
        StringBuilder Path = null;
        mcp_module = MCP2210.M_Mcp2210_OpenByIndex(base_vid, PID, 0, Path);

        //швидкість передачі даних
        uint rate = 9600;
        //налаштування виводу CS
        uint cs_idle = 0xFFF;
        uint cs_activ = 0xFBF;
        //налаштування затримок
        uint cs_to_data_dly = 0;
        uint data_to_data_dly = 0;
        uint data_to_cs_dly = 0;
        //кількість байт прийому/передачі
        uint numberbyte = 3;
        //режим роботи SPI
        byte spi_mode = (byte)MCP2210.M_MCP2210_SPI_MODE0;
        //масиви прийому/передачі
        byte[] Tx = new byte[3];
        byte[] Rx = new byte[3];
        //режим роботи портів 0 - gpio, 1 - spi, 2 - alter
        byte[] gpio = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00 };
        uint run_state_pin = 0;
        uint dir_pin_state = 0;

        //налаштування модуля
        MCP2210.M_Mcp2210_SetGpioConfig(mcp_module,
(byte)MCP2210.M_MCP2210_VM_CONFIG, gpio, run_state_pin, dir_pin_state,
                    (byte)MCP2210.M_MCP2210_REMOTE_WAKEUP_DISABLED,
(byte)MCP2210.M_MCP2210_INT_MD_CNT_NONE,
                    (byte)MCP2210.M_MCP2210_SPI_BUS_RELEASE_DISABLED);
                MCP2210.M_Mcp2210_SetSpiConfig(mcp_module,
(byte)MCP2210.M_MCP2210_VM_CONFIG, ref rate, ref cs_idle, ref cs_activ, ref
cs_to_data_dly, ref data_to_data_dly, ref data_to_cs_dly, ref numberbyte, ref
spi_mode);
        //додавання до списку підключених
        AddPid(PID);
        return "Module connect PID " + (PID.ToString());
        }
```

MCP2210.M_Mcp2210_GetConnectedDevCount – функція яка входить до стандартної бібліотеки, та необхідна для визначення наявності модуля з відповідною PID та VID адресую. У разі наявності модуля та відсутності первинного налаштування програма записує у змінні необхідні параметри, та передає їх завдяки функції MCP2210.M_Mcp2210_SetGpioConfig(…).

Таблиця 3.1 – Опис основних функцій які входять у MCP 2210.DLL

Назва функції	**Опис**
Mcp2210_OpenByIndex	Відкриває з’єднання з пристроєм MCP2210, ідентифікованим за даним VID:PID й номер індексу і повертає дескриптор файлу.
Mcp2210_GetConnectedDevCount	Надає кількість підключених пристроїв із заданим VID:PID.
Mcp2210_Close	Спробує закрити з'єднання з MCP2210.
Mcp2210_Reset	Скидає USB-пристрій.
Mcp2210_SetGpioPinDir	Встановлює напрямок GPIO.
Mcp2210_GetGpioPinDir	Забезпечує поточний напрямок контакту GPIO. Значення дійсні лише для контактів раніше налаштовані як GPIO.
Mcp2210_GetGpioPinVal	Надає поточні значення GPIO. Значення дійсні лише для контактів раніше налаштовані як GPIO.
Mcp2210_SetGpioPinVal	Встановлює поточні значення GPIO. Також повертає поточні значення.
Mcp2210_GetGpioConfig	Забезпечує поточну конфігурацію GPIO або конфігурацію GPIO для ввімкнення за замовчуванням (NVRAM).
Mcp2210_SetGpioConfig	Встановлює поточну конфігурацію GPIO або конфігурацію GPIO за замовчуванням для включення живлення (NVRAM). Якщо мікросхема NVRAM захищена паролем, конфігурація GPIO не може бути виконана, API повертає E_ERR_BLOCKED_ACCESS.

Продовження таблиці 3.1

Назва функції	Опис
Mcp2210_GetSpiConfig	Отримує параметри передачі SPI для поточної (VM) конфігурації.
Mcp2210_SetSpiConfig	Налаштовує параметри передачі SPI для поточної (VM) конфігурації.
Mcp2210_GetSpiStatus	Якщо немає помилки зв’язку пристрою, ця команда повертає 0 (E_SUCCESS) і оновлює покажчики інформацією про стан передачі SPI.

3.3 Розробка програмного забезпечення пристрою керування

Як зазначалось вище для керування установкою необхідно отримати координати сонця та інформацію про погоду. Для отримання інформації про стан погоди використовується WEB сервіс OpenWeather. Цей сервіс дозволяє отримати всю необхідну інформацію відносно координат місцевості. API сервісу потребує наступного виклику:

https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}. Де lat широта, lon довгота, API key - унікальний ключ користувача, для отримання ключа необхідно зареєструватися на сервісі. Сервер повертає інформацію відповідно структури на рисунку 3.2.

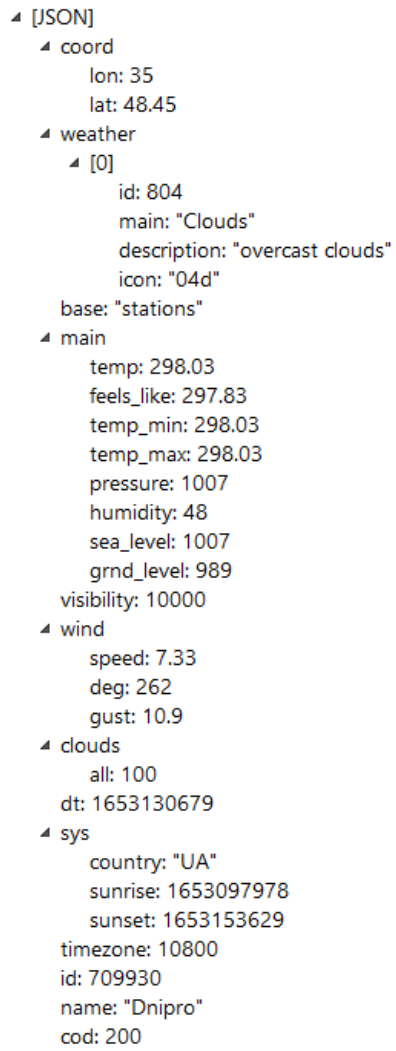


Рисунок 3.2 – Структура отриманого повідомлення с серверу

Нам необхідно в цьому древі лише категорія weather. В якій знаходиться стан погоди та її більш детальний опис. Функція яка забезпечує зчитування цих параметрів виглядає наступним чином:

```
string url_weather_sourse =
"https://api.openweathermap.org/data/2.5/weather?lat=48.45&lon=35&appid=275731b3
3265047fd805e30b9ffac55e";

HttpWebRequest httpWebRequest =
(HttpWebRequest)WebRequest.Create(url_weather_sourse);
HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();

using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
{
    responsWeather =
JsonConvert.DeserializeObject<Rootobject>(streamReader.ReadToEnd());
}
```

Як видно використовується бібліотека Newtonsoft.Json, для десеріалізації отриманих даних, які записується в об’єкті C#, оглянути їх можна у відповідному додатку.

Розрахунок положення сонця відбувається за описаною в першому розділі цієї роботи методом. Код розрахунку представлений у додатку функцією SolarCoordinates, а параметри змінних у таблиці 3.2.

Таблиця 3.2 – Тип та опис змінних

Назва змінної	**Тип**	**Опис**
LONGITUDE	const double	Довгота місця встановлення трекеру.
LATITUDE	const double	Широта місця встановлення трекеру.
sun_declination_table	double[,]	Двох мірний масив в якому записані дані нахилу сонця протягом року.
EOT_time	double	Значення рівняння часу.
EOT_variable	double	Додаткова змінна для розрахунку.
true_solar_time	double	Істинний сонячний час.
UTC_time	double	Час у форматі UTC, виражений у хвилинах.
astronomical_time	double	Астрономічний час.
high_solar	double	Висота сонця.
azimuth_solar	double	Азимут сонця.
day_year	int	День року.
current_time	DateTime	Системний час.

Таблиця 3.3 представляє собою критично важливі функції для роботи об’єкта. А таблиця 3.4 їх опис.

Таблиця 3.3 – Основні функції для роботи об’єкта

Назва функції	**Тип**	**Вхідні параметри**	**Вихідні параметри**
StepMotorControl	string	- set_angle кут який необхідно встановити - cur_angle кут який є зараз - mcp_module індефікатор модуля	Повідомлення про результат роботи
ADCDataReception	void	- mcp_module індефікатор модуля - data_value масив для повертання параметрів зчитвуання	-
ControlPorts	void	- mcp_module індефікатор модуля - status_port увімкнути чи вимкнути порт - cod_port номер порту	-
UsbConnect	string	- mcp_module індефікатор модуля для повернення - PID адреса модуля	Повідомлення про результат роботи
SolarCoordinates	void	-	-
AutoControl	void	-	-
TimerGetWither_Tick	void	подія	-
TimerRealTime_Tick	void	подія	-

Таблиця 3.4 – Призначення основних функцій

Назва функції	**Призначення**
StepMotorControl	Керування кроковим двигуном за реалізацією деяких противо аварійних заходів.
ADCDataReception	Зчитування праматерів з датчиків.
ControlPorts	Управління портами MCP 2210.
UsbConnect	Підключення до модулю MCP 2210 та його конфігурування.
SolarCoordinates	Розрахунок положення сонця.
AutoControl	Функцію виконує автоматичне керування.
TimerGetWither_Tick	Подія таймеру для отримання свідчень про погоду.
TimerRealTime_Tick	Подія таймеру для реалізації роботи системи с заданими проміжками часу.

Важливим процесом є зчитування параметрів. Нижче представлена частина коду для зчитування параметру напруги.

```
////////////////////////////////////////////////////////////////////////////////
                Tx[1] = 0b10000000;

                MCP2210.M_Mcp2210_xferSpiData(mcp_module, Tx, Rx, ref rate, ref
numberbyte, 0x80000000);
                Rx[0] = Rx[2];
                Rx[1] = Convert.ToByte(Rx[1] & 0b00001111);
                temp = BitConverter.ToInt16(Rx, 0);

                volteg = temp * 0.0006103515625 * voltage_factor;

////////////////////////////////////////////////////////////////////////////////
```

Перед зчитуванням параметрів необхідно підготовити масив для відправки, у ньому зберігається інформація про канал з якого необхідно зчитувати значення напруги. Рисунок 3.3 відображає ці біти Singl/Diff ,D1,D0. D2 не використовуються у 4 канальній версії АЦП. Ці данні записуються у масив Tx та передаються до команди M_Mcp2210_xferSpiData. Функція поверне масив байтів Rx, у якому і знаходяться зчитані дані. Їх необхідно вирівняти по лівому краю та перетворити у ціле число, яке може змінюватися від 0 до 4095.

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2*	D1	D0		
1	X	0	0	single ended	CH0
1	X	0	1	single ended	CH1
1	X	1	0	single ended	CH2
1	X	1	1	single ended	CH3

Рис 3.3 – Контролюючі біти

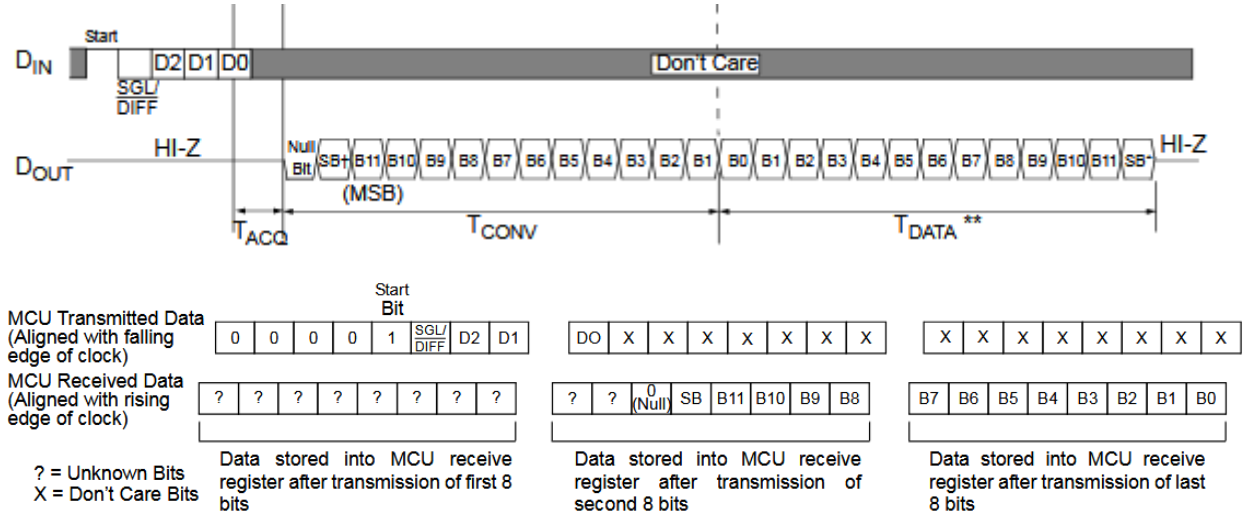


Рис 3.4 – Структура повідомлень для MCP 3302

Число записане в змінну temp необхідно привести до типу вимірюємої величенні. Яка є добутком temp на коефіцієнт LSB (3.1) та коефіцієнту калібрування.

$$LSB_{Size} = \frac{2 * V_{ref}}{8192} \tag{3.1}$$

Де Vref – напруга опорного джерела живленя.

Vref становить 2.5В. Звідси:

$$LSB_{Size} = \frac{2 * 2.5}{8192} = 0.00061035$$

Для управління необхідно вміти керувати портами GP модуля MCP 2210, для вирішення цієї задачі використовується функція ControlPorts. Фрагмент коду для GP 4 зазначено нижче:

```
////////////////////////////////////////////////////////////////////////////////
else if (cod_port == 104)
            {
                if (status_port == 1 && port_arr_controll[3] == false)
                {
                    on_off_bit_port += 16;
```

```
                port_arr_controll[4] = true;
            }
            else if (status_port == 0 && port_arr_controll[3] == true)
            {
                on_off_bit_port -= 16;
                port_arr_controll[4] = false;
            }
        }
////////////////////////////////////////////////////////////////////////////////
```

Головною цілю є керування декількома портами, вмикати одні виводи не впливаючи на інші. Для цього в модуль відправляється число, ідентифікатор активних портів, таблиця 3.5. Для того щоб увімкнути необхідні порти передається число, наприклад 52, це увімкне порти які є сумою цього числа. Зеленим кольором вказані увімкнуті порти при передачі числа 52.

Таблиця 3.5 – Кодування портів для керування

GP0	GP1	GP2	GP3	GP4	GP5	GP6	GP7
1	2	4	8	16	32	64	128

Якщо ж необхідно вимкнути порт, то потрібно відняти індекс порта від загальної суми. Ця сума передається до функції M_Mcp2210_SetGpioPinVal. Слід зазначити що порт GP8 є особливим випадком, його не можливо використовувати як CS, бо при початку передачі флаг цього порта скидається апаратно. Це виходить із організації внутри схемної логіки MCP 2210 оскільки для GP8 використовується окремий байт. Виробник не рекомендує використовувати його і в якості дискретного виходу, бо він також скидається при початку передачі по SPI.

Для керування використовуються ланцюги функцій. На рисунку 3.5 вказаний алгоритм роботи автоматичного режиму, а саме функції AutoControl. Вона є другою в ланцюгу після таймеру, який забезпечує її роботу та контроль. Словесний опис наступний, перевірка на наявність сонця. Якщо воно сіло за горизонт, то виконується поворот на початкову позицію, якщо ж ні то переходить до списку умов. Перші дві умови забезпечують не рухоме положення установки при азимуту поза діапазоном нахилу. Та утримують установку певний час у

крайньому положенні. Коли азимут досягне робочого діапазону буде виконуватися перевірка на погодні умови та уразі їх відсутності установка повернеться відповідно азимуту. Сніг має додаткове розгалуження, в якому обирається найкоротший напрямок досягнення кінцевого кута.

Рис 3.5 – Алгоритм логіки роботи автоматичного режиму

3.4 Розробка програмного забезпечення людино-машинного інтерфейсу

Для роботи людини необхідно створити людино-машинний інтерфейс. Він повинен не тільки забезпечувати комфортну роботу, ай забезпечити відповідність з схемою інформаційних потоків. Для виконання цих умов використовуються компоненти зображенні на рисунку 3.6. А їх призначення наведене у таблиці 3.6.

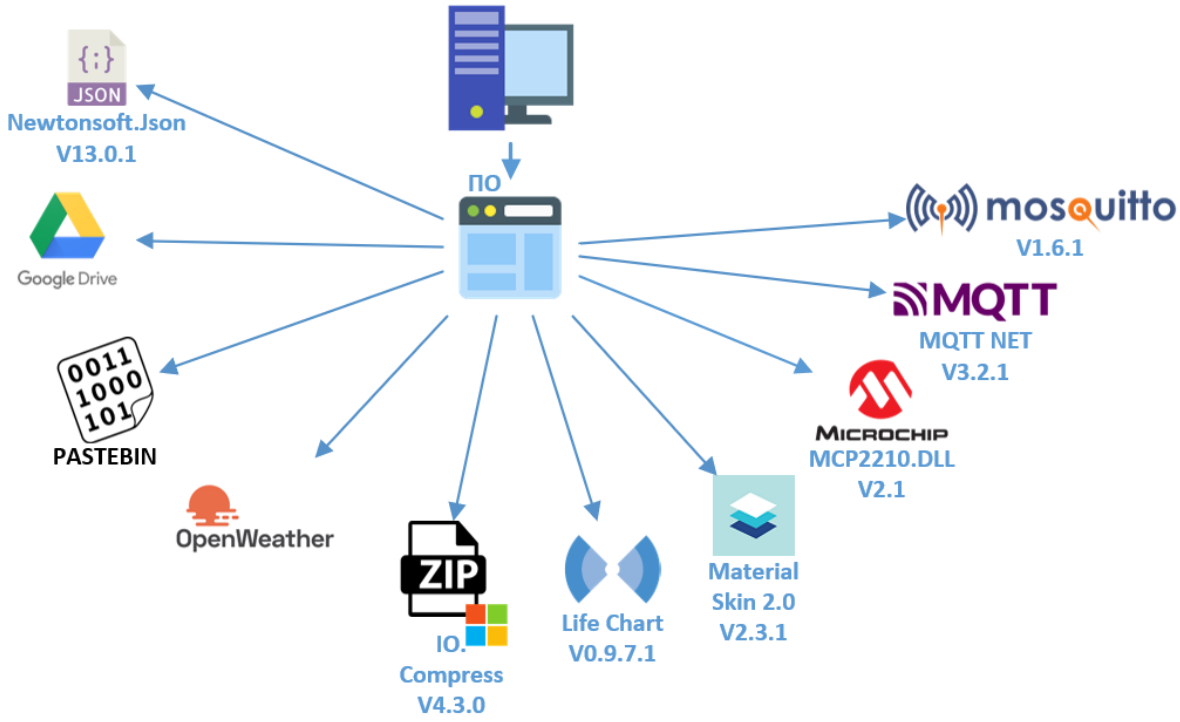


Рис 3.6 – Перелік бібліотек та сервісів задіяних у системі

Таблиця 3.6 – Призначення бібліотек

Назва бібліотеки або сервісу	**Призначення**
Newtonsoft.Json	Необхідний для роботи з Json файлами, їх серіалізції та десіаріалізцаї.
Google Drive API	Сервіс який використовується для оновлення програмного забезпечення.

Продовження таблиці 3.6

Назва бібліотеки або сервісу	**Призначення**
PASTEBIN	Сервіс для перевірки наявності оновлень.
OpenWeather	Сервіс для отримання погоди.
Microsoft.IO.Compress	Бібліотека для роботи з ZIP архівами.
Life Charts	Бібліотека для роботи з графіками.
Material Skin 2.0	Дозволяє створити додаткові елементи користувача.
MCP2210	Необхідна для роботи з модулем управління.
MQTT.NET	Реалізує необхідні функції клієнта.
Mosquito	Сервер для організації локального зв’язку.

Як видно програмне забезпечення пристрою керування та людино-машинного інтерфейсу це одне ціле. Та розроблено на мові програмування C# з використанням WinForms (Рис 3.7). Класи головної форми на рисунку 3.8.

Рис 3.7 – Структура форм

Рис 3.8 – Класи головної форми

Головне вікно програми (Рис 3.11) дає оператору всю необхідну інформацію та дозволяє керувати об’єктом. З лівої сторони знаходиться схематична зображення установки у який змінюється нахил сонячного елементу відповідно до реального об’єкту, над ним знаходиться іконка стану погоди їх варіанти зображені на рисунку 3.9.

Рис 3.9 – Варіанти іконок про стан погоди

Слайдер знизу дозволяє змінювати кут нахилу оператором, у разі обрання автоматичного режиму він стає не активним.

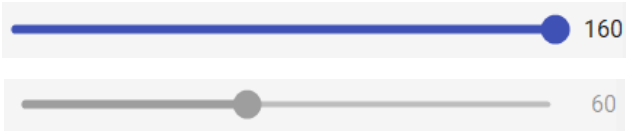


Рис 3.10 – Режими слайдеру

Перемикачі нижче дозволяють увімкнути автоматичний режим, та режим дистанційного керування.

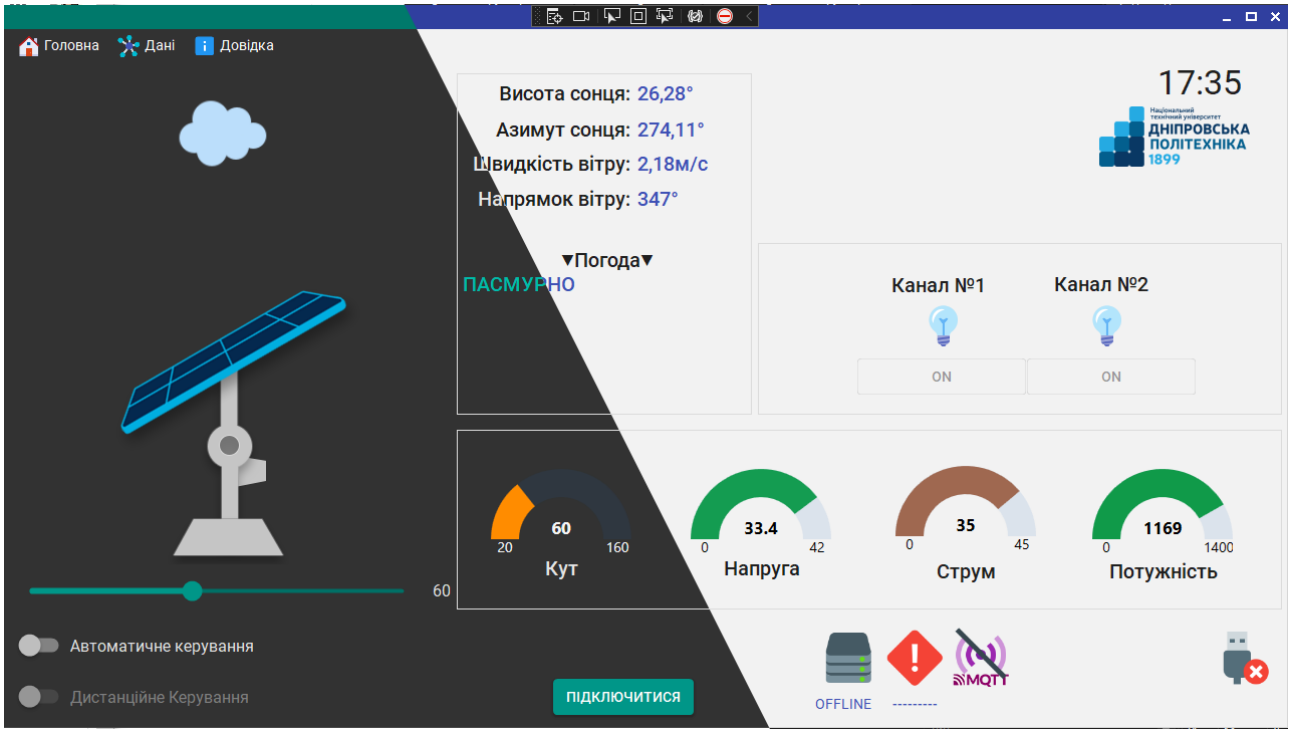


Рис 3.11 – Вигляд головного вікна у двох темах

Вікно с параметрами відображає результати розрахунків положення сонця, й більш детальну інформацію про стан погоди. Нижче знаходиться параметри виміряних величин та блок індикаторів статусу рисунок 3.12.

Рис 3.12 – Блок індикаторів

Для налаштування програми необхідно натиснути на «Головну» та обрати пункт «Налаштування». В цьому вікні можна змінити коефіцієнти виміряємої величини, змінити сервер та проміжки часу для запитів, вимкнути формування звітів, їх форму та періодичність запису. Вимкнути ведення системного журналу та міняти параметри MQTT.

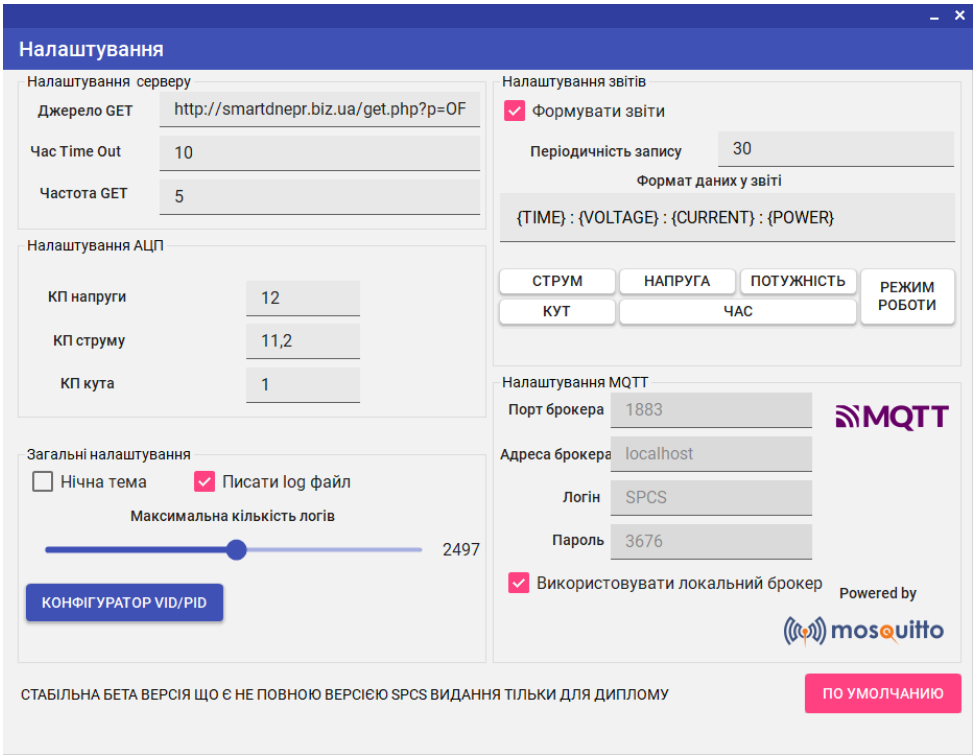


Рис 3.13 – Вікно налаштування

Робота з даними. Представлена у пункті «Данні». Вікно виглядає наступнім чином (Рис 3.14). Тут можна переглянути параметри у реальному часі, графікі оновлюються кожну секунду, та можна вимкнути не потрібні параметри для більш конкретного перегляду. У правому лівому вуглу знаходяться повзунки для зміни зуму, тобто довжини графіку у екрані та розміру буферу максимальне значення якого 1000 точок для кожного графіку. Графік автоматично масштабується по вертикальній вісі. Натиснув кнопку «Стоп» можна зупинити графік та на вестись на лінії графіку для отримання таблиці значень. Поле можна пересувати мишкою, що дозволить переглянути значення, які за межами екрану.

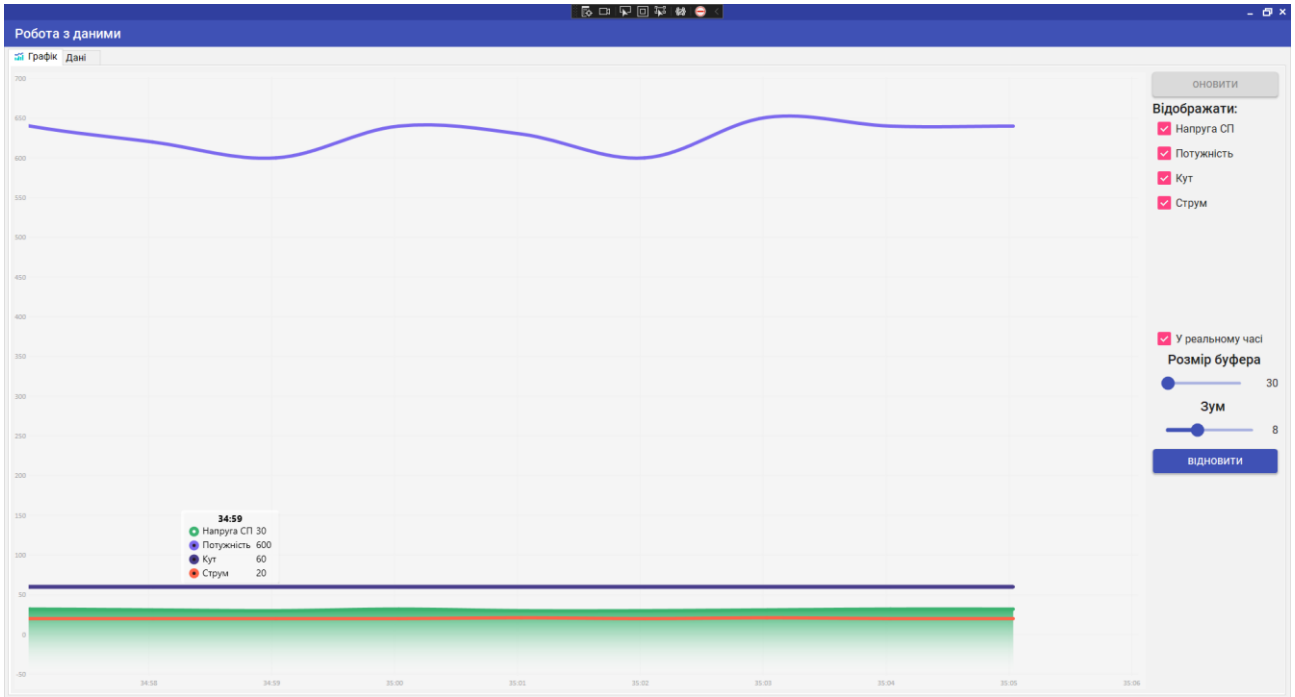


Рис 3.14– Вигляд вікна у режимі реального графіку

Другий режим роботи це статичний графік, кількість точок які відображаються технічно обмеженні 30 000 тисячами. Для оновлення графіку необхідно натиснути відповідну кнопку. Функціонал той же але є можливість збільшувати зображення поворотом колеса мишки. У разі використання більше 4000 точок на екрані обсяг займаємо оперативної пам’яті збільшується на 480 мб. Нижче наведений рисунок 3.15 графіку, який був заповнений випадковим числом, як видно для збільшення продуктивності точність малювання

знижується. Коли кількість точок занадто велика, але при приближені якість збільшується.

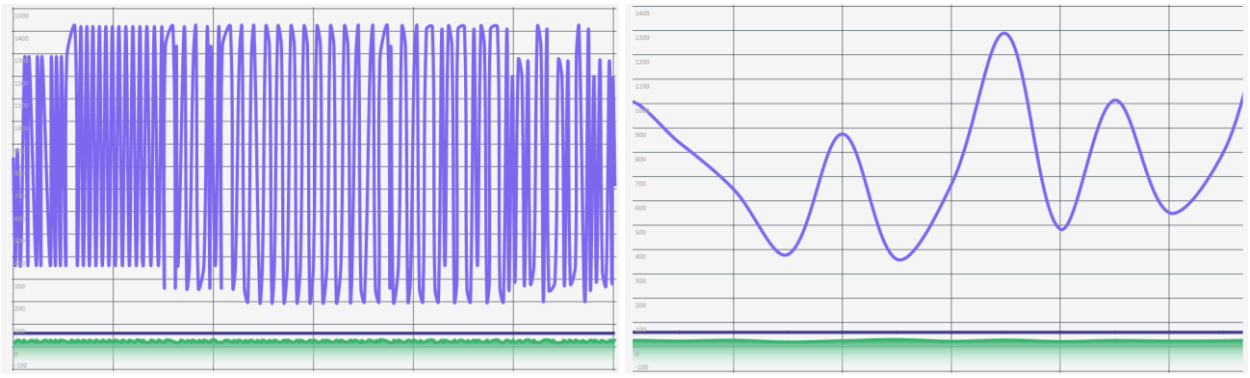
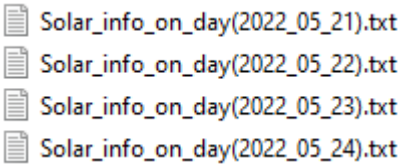


Рис 3.15– Графік з 2 тисяч точок загальний та масштабований вид

Архівування параметрів відбувається завдяки запису обраних даних у файл формату TXT або CSV. Формат запису може бути любим, він обирається в налаштуваннях програми. Якщо ця функція не потрібна її можна вимкнути. Програма сама створює файл та записує туди необхідну інформацію, назва файлу типова, та складається з заголовку та дати дня.




```
{TIME} : {VOLTAGE} : {CURRENT} : {POWER}
12.17 : 33.4 : 35.0 : 1169
12.18 : 33.4 : 35.0 : 1169
12.18 : 33.1 : 35.0 : 1159
12.19 : 33.5 : 35.0 : 1173
12.19 : 33.4 : 35.0 : 1169
12.20 : 33.4 : 35.0 : 1169
12.20 : 33.1 : 35.0 : 1159
12.21 : 32.9 : 34.9 : 1148
12.21 : 33.0 : 35.0 : 1155
12.22 : 33.1 : 35.0 : 1159
```

Рис 3.16 – Приклад архівування за замовчуванням

Організація відображення помилок відбувається не тільки піктограмами, а й спливаючими повідомленнями та логуванням. Повідомлення можуть бути двох видів. Перший, це не великий прямокутник, який з’являється знизу вікна та з

часом самостійно зникає. Другий тип це банер, який присутній весь час. Він затемнює весь інтерфейс та очікує дії оператору. Поява обох повідомлення відбувається з звуком. Логування (Рис 3.18) здійснюється записом у текстовий файл, повідомлення та його типу. Info, error, fatal. Розмір логу встановлюється в налаштуванні. При досягненні заданої кількості, програма автоматично зітре перші 20 записів.

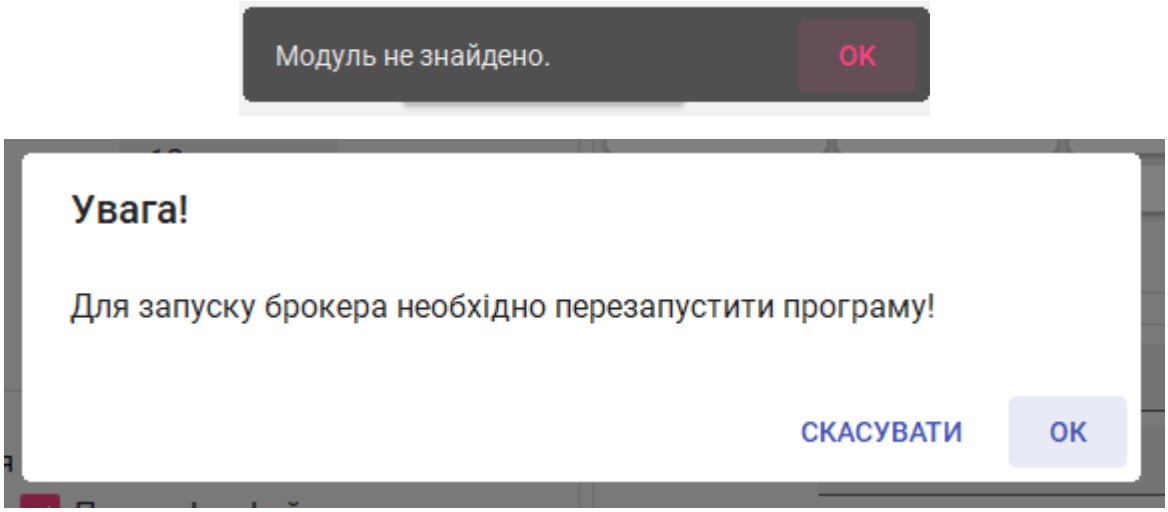


Рис 3.17 – Два виду повідомлення

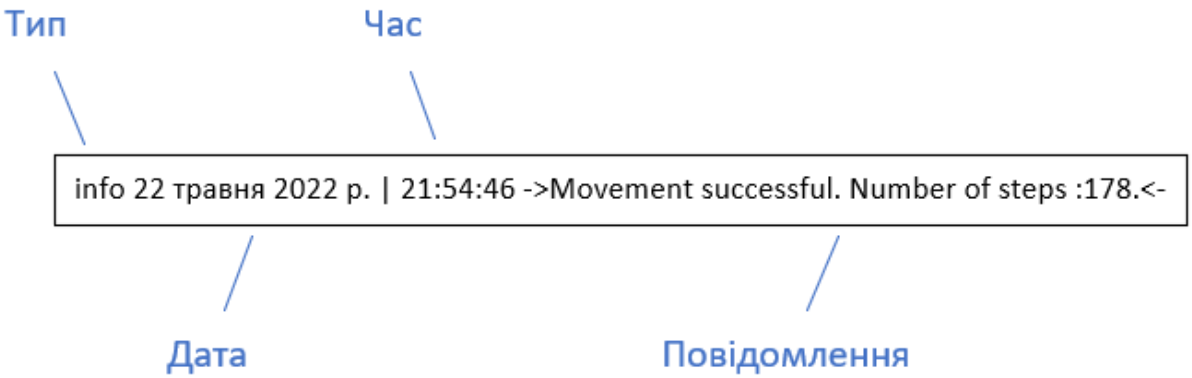


Рис 3.18 – Структура логу

Реалізація функції клієнту забезпечується завдяки GET запиту та MQTT клієнту. Це дозволяє моніторити систему на відстані та впливати на деякі її параметри. З веб-сайту або з мобільного додатку (рис3.21), в якості брокеру може виступати любий веб сервіс або локальний брокер, завдяки Mosquito.

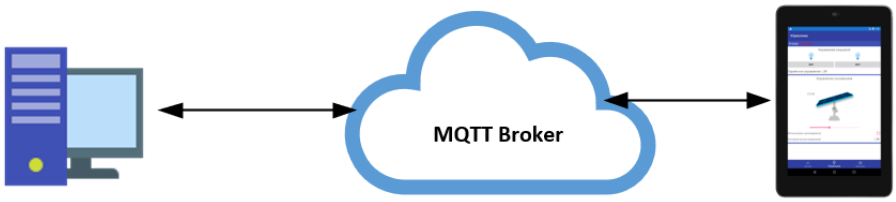


Рис 3.19 – Структура обміну даними

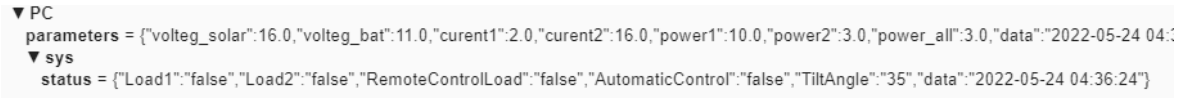


Рис 3.20 – Структура розгалужень

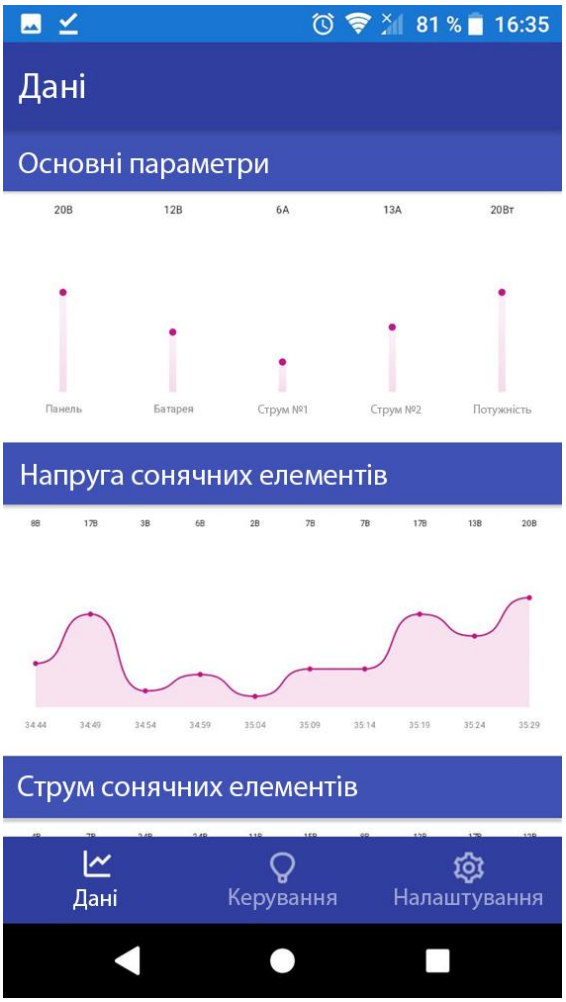


Рис 3.21 – Головне вікно програми

3.5 Висновки за розділом

В результаті роботи було створене відповідне програмне забезпечення, яке надає змогу реалізувати не обхідні функції для роботи об’єкту. Надати оператору

можливість отримувати всю необхідну інформацію та впливати на роботу системи. Програмне забезпечення повністю відповідає вимогам попередніх розділів.

4 ЕКОНОМІКА

4.1 Розрахунок капітальних витрат

Сонячний трекер допомагає збільшити ефективність здобування енергії, а отже збільшити прибуток від її продажу або знизити витрати на спожиту енергію. Функції які були реалізовані у попередніх розділах дозволять зменшити кількість робітників та спец техніки, а отже знизити експлуатаційні витрати. Також система дуже легко масштабується, шо дозволить наростити кількість установок без суттєвих змін архітектури та необхідності навчання персоналу.

Для розрахунку капітальних витрат необхідно визначити ціну обладнання та усіх робіт по її встановленню. Оскільки використовується власноруч розроблена плата управління, то необхідно розрахувати її вартість у спрощену вигляді. Основні компоненти та їх ціна у таблиці 4.1.

Таблиця 4.1 – Ціна основних елементів модуля

№	Найменування	Кількість	Ціна за одиницю, грн	Сума, грн
1	TB6600HG	1	635	635
2	MCP3302	1	325	325
3	MCP2210	1	200	200
4	MCP1525	1	50	50
5	ACS714	1	386	386
6	LM358	2	4	8
7	L7805ABV	1	6	6
			Всього	1 610

Додаткові компоненти такі як, роз’єми та резистори тощо. Становлять додатково ще 400 грн, а ціна виготовлення однієї плати на виробництві сягає 30 грн. З урахуванням логістики загальна ціна модуля 2100 грн.

Ціна всього обладнання взяті з роздрібних магазинів та представлені в таблиці 4.2, з урахуванням ПДВ.

Таблиця 4.2 – Ціна усього обладнання

№	Найменування	Кількість	Ціна за одиницю, грн	Сума, грн
1	Beelink Z83II	1	2900	2900
2	C&T Solar CT60320-M	4	3200	12800
3	Крокoвий двигун NEMA34	1	1570	1570
4	Редуктор RV40-86	1	3600	3600
5	AXIOMA ISPWM 2000	1	10260	10260
6	Mean Well 156	1	704	704
7	Kubler IN 81	1	882	882
8	MCP 2210 Modul	1	2090	2100
			Всього	34 816

Витрати на доставку та встановлення без урахування модуля MCP 2210 становлять 10% від вартості обладнання. Та сягає 3 271,6 грн. Установка не потребує налагодження і почне працювати після встановлення всього обладнання та програмного забезпечення. Капітальні витрати визначаються за формулою 4.1.

$$К_{пр} = К_{об} + З_{тзс} + З_{м} + З_{н} + З_{пр} \quad (4.1)$$

Де $К_{об}$ – вартість обладнання, $З_{тзс}$– транспортно-заготівельні і складські витрати, $З_{м}$ – витрати на монтажні роботи, $З_{н}$ - витрати на налагоджувальні роботи, $З_{пр}$– інші одноразові вкладення грошових коштів. Тому капітальні витрати становлять:

$$К_{пр} = 34\,816 + 3271{,}6 + 0 + 0 = 38\,087{,}6 \text{ грн}$$

4.2 Розрахунок експлуатаційних витрат

В результаті використання установки та її обслуговування виникають експлуатаційні витрати, які знаходяться за формулою 4.2.

$$С = С_а + С_з + С_с + С_т + С_є + С_{пр} \text{ ,грн.} \quad (4.2)$$

де $С_а$ амортизаційні відрахування, $С_з$ заробітна плата обслуговуючого персоналу, $С_с$ єдиний соціальний внесок.

$С_т$ витрати на технічне обслуговування й поточний ремонт устаткування та мереж, $С_є$ вартість електроенергії, що буде споживана об'єктом проектування або втрат електроенергії, $С_{пр}$інші експлуатаційні витрати.

4.2.1 Розрахунок амортизаційних відрахувань

Оскільки обладнання зношується з часом необхідно обдумати створення фонду для амортизації витрат у разі оновлення установки або капітальних ремонтів. Конструкція установки має великий запас по міцності, тому період експлуатації всієї системи дорівнюю періоду оптимальної експлуатації сонячних елементів і становить 10 років. За наявності якісного обслуговування.

Річний фонд амортизації визначається за формулою 4.3. Т – це період експлуатації.

$$А = \frac{1}{Т} \cdot 100\% \quad (4.3)$$

Результати амортизаційних відрахувань наведена у таблиці 4.3

Таблиця 4.3 – Таблиця результату розрахунку фонду амортизації

№	Найменування	Капітальні інвестиції, грн.	Норма амортизації, %	Сума амортизації, грн.
1	Beelink Z83II	2900	10	290
2	C&T Solar CT60320-M	12800	5	640

Продовження таблиці 4.3

№	Найменування	Капітальні інвестиції, грн.	Норма амортизації, %	Сума амортизації, грн.
3	Крокoвий двигун NEMA34	1570	10	157
4	Редуктор RV40-86	3600	10	360
5	AXIOMA ISPWM 2000	10260	10	1026
6	Mean Well 156	704	10	71
7	Kubler IN 81	882	5	45
8	MCP 2210 Modul	2100	10	210
			Всього	2 799

4.2.2 Розрахунок річного фонду заробітної плати

Установка не потребує наявності людини на робочому місці. Але для організації обслуговування та реагування на не стандартні ситуації, повинен бути черговий інженер. В його обов'язки входить періодичне планове обслуговування об'єкту, та контроль параметрів установки. Тому необхідно розрахувати заробітну плату для співробітника, та річний фонд робочого часу. Робітник працює 4 годи в день, 5 днів на неділю. Кількість вихідних днів з урахуванням свят становить 116. Тому фонд робочого часу становить:

$$F_H = (Д_к - Д_{св} - Д_{вих}) * T_{зм} = (365 - 116) * 4 = 996 \text{ год.}$$

Таблиця 4.4 – Річний фонд заробітної плати

Найменування професій робітників	Годинна тарифна ставка грн.	Номінальний річний фонд робочого часу	Усього основна зарплата, грн
Черговий інженер	85	996	84 660

Додаткова заробітна плата, це преміальні або понаднормові виплати, та визначаються як 8% від основної заробітної плати. Й становить 6 773 гривні.

Загальний фонд заробітної плати становить:

$$С_з = З_{осн} + З_{дод} = 84\,660 + 6\,773 = 91\,433 \text{ грн.}$$

4.2.3 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% від заробітної плати. Отже його розмір буде становити 20 156 грн.

4.2.4 Визначення річних витрат на технічне обслуговування і поточний ремонт

Витрати на обслуговування становлять близько 4% від капітальних витрат , тому їх розмір становить:

$$С_{р.т.о.} = 38816 * 0{,}04 = 1392{,}64$$

4.2.5 Розрахунок вартості спожитої електроенергії

Розрахунок вартості спожитої енергії не доцільний для цього об’єкту, адже він є виключенням, бо кількість генеруємої електричної енергії більша за ту, що споживається.

4.2.6 Визначення інших витрат

Для організації завдань з охорони праці необхідно виділити кошти на спец одяг, та організацію інструктажів з безпеки роботи. Прийнято вважати, що ці витрати не перевищують 4% від річного фонду заробітної плати. Тому витрати будуть становити:

$$С_{інш} = 84\,660 * 0.04 = 3386{,}4 \text{ грн}$$

Визначивши всі види витрат можемо розрахувати експлуатаційні витрати за формулою 4.2, одна людина може обслуговувати 30 установок, тому витрати пов’язані з працівником можна скоротити у 30 разів.

$$С = С_а + С_з + С_с + С_т + С_є + С_{пр} =$$

$$= 2\,799 + 3047{,}8 + 671{,}8 + 1392{,}64 + 0 + 112{,}88 = 8\,109{,}7 \text{ грн}$$

4.3 Висновки

Метою автоматизації є збільшення ефективності здобування енергії, яка є кінцевим продуктом електростанції. Зменшення займаних площ, а отже і витрат на їх купівлю або оренду. Трекер даної конструкції дозволяє збільшити видобутку енергії на 35%, в літку на 37% (рис 4.1). А завдяки повністю автоматичному керуванні не потребує окремого фахівця. Додаткові системи дозволяють знизити витирати на спецтехніку та на оплаті персоналу. Наприклад система автоматичного скидання снігу.

Станом на 2022 рік вартість одного кіловату електроенергії по зеленому тарифу становить 0,163 євро. В середньому протягом року 4 модуля потужністю 320Вт дадуть 43,3 євро прибутку за місяць роботи. Звідси річний прибуток становить приблизно 519,6 євро що при курсу 1€ = 31,57грн, становить 16 398,73 грн на рік. Чистий прибуток сягає розміру 8 289.03 грн на рік.

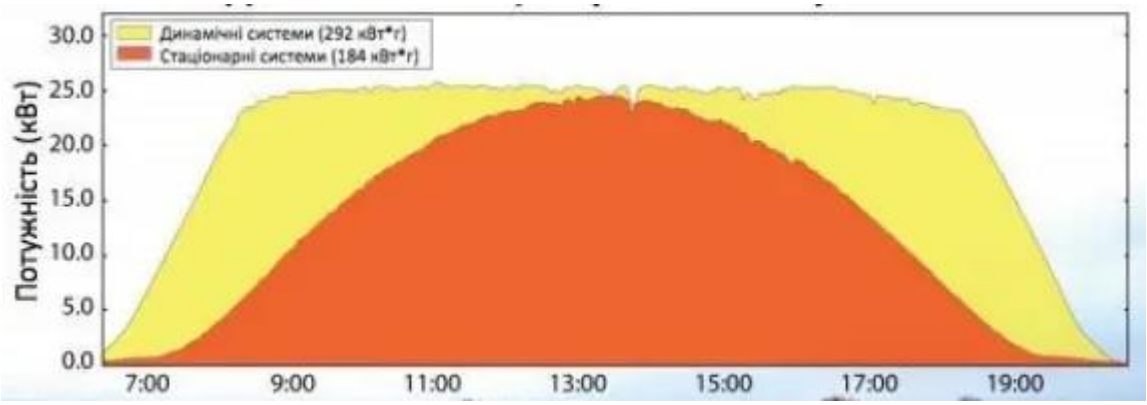


Рис 4.1 – Ефективність рухомої системи

В той час як не рухома система мала показник на 35% нижчий. А отже ця система збільшує річний прибуток на 5739.56 грн. Без урахувань економії на землі. Період повернення затрат на установку сягає 4,43 роки, при періоді експлуатації в 10 років. З цього можна зробити висновок що використання трекеру є економічно вигідним та доцільним.

5 ОХОРОНА ПРАЦІ

5.1 Аналіз небезпечних і шкідливих виробничих чинників сонячної фотоелектричної установки на поворотному механізмі

Сонячний трекер який розроблений у цій роботі забезпечує позиціонування сонячних елементів. Та встановлюється на відкритій місцевості, яка далека від високих об’єктів, які можуть створювати тінь. Але дана установка має декілька небезпечних чинників, оскільки конструкція повністю металева та має шафу с інвертором, у якому присутня напруга 220 вольт. Є ризик враження електричним струмом. Лінії високої напруги проходять через кабель канал, прокладений у землі. Навіть відключена система продовжує надавати електроенергію під дією сонця і може спричинити до:

- Термічних опіки
- Пошкодженню м’язів, нервів і тканин
- Падіння від шоку
- Летальних випадків

Сонячні пеналі генерують небезпечний рівень постійного струму. У разі ураження відбувається постійне скорочування м’язів, що ускладнює розрив контакту з об’єктом, який знаходяться під напругою.

Трекер має рухомі вузли та механізми, тому під час обслуговування та недотримання правил безпеки можна отримати травму, конструкція поворотної рами має малу площину профілю, а двигун створює зусилля у 600 кг. Цієї сили достатньо щоб нанести тяжкі травми.

Верхня точка кріплення обертаючого металевого каркасу, знаходиться на висоті 3 метра, а обслуговування може відбуватись у різні погодні умови, та с різними температурами. Це створює додаткові ризики, які приводять до падіння з висоти та драбини, або можливого підсковзування. У результаті чого можуть виникнути:

- Переломи або розтягнення
- Колоті ушкодження

- Травми спини, шиї та голови
- Порізи та синці
- Внутрішні травми

Робота на відкритому повітрі протягом тривалого часу в жаркому та вологому середовищі, може привести до теплових захворювань, зневоднення, втоми. Робітник також може відчувати відблиски від поверхонь фотоелектричних модулів під час роботи в день. Відблиски бувають прямими або непрямими і здатні погіршити зір. Також це впливає на продуктивність роботи, ускладнення процесу встановлювання модулів й на безпеку установки. Адже монтаж кріплення болтів, гайок, навісного обладнання, візуально складні завдання. Працівник піддається тривалому впливу сонячного випромінювання, яке посилюється від відбиття сонячними панелями. Результатами стають захворювання: катаракта, меланома, сонячні опіки, деградація шкіри.

Під час обслуговування може з’явитися потреба у замінні або знятті фотоелектричного модуля, які мають велику вагу та невдалі розміри для утримання в руках, особливо панелі великої потужності більше 300 Вт. Які використовуються на даній установці. Неправильний підйом може спричинити розтягнення та серйозні травми спини, такі як:

- Грижа міжхребцевих дисків
- Розрив ротаторної манжети
- Розтягнення стегон і нижньої частини спини

5.2 Інженерно-технічні заходи з охорони праці

До обслуговування установки допускаються особи не молодше 18 років, яка повинна мати повну середню освіту та професійно-технічну освіту, без вимог до стажу роботи або повну загальну освіту, та професійну підготовку на виробництві. Працівник має мати спец одяг та додаткове обладнання, а саме: робоча роба під сезон роботи, захисні рукавиці для захисту від ураження струмом, рація для зв’язку з оператором у разів виникнення аварійної ситуації.

Усе обладнання обов’язково заземлене, це не тільки вимога з охорони праці, ай необхідність захисту обладнання та інформаційних каналів від грози та екранування від електромагнітних перешкод. Експлуатація установки без заземлення забороняється. Заземлення установки рекомендується проводити на місці, у разі не можливості, використовується окремий кабель, перетин якого більший у 1,5 разів за фазову жилу. Все обладнання встановлюється у металеві герметичні бокси з рівнем захисту не нижче IP 65. Захист установки здійснюється завдяки пристрою захисного відключення, який встановлюється на виході інвертора та на вході державної мережі, у розподільчому пункті.

Забороняється використовувати дві металеві балки конструкції в якості заземлення. Адже їхній супротив виходить за межі рекомендованого значення у 4 Ом для 220В. Замість цього використовується окремий заземлюючий контур. Схема підключення обладнання представлена на малюнку 5.1.

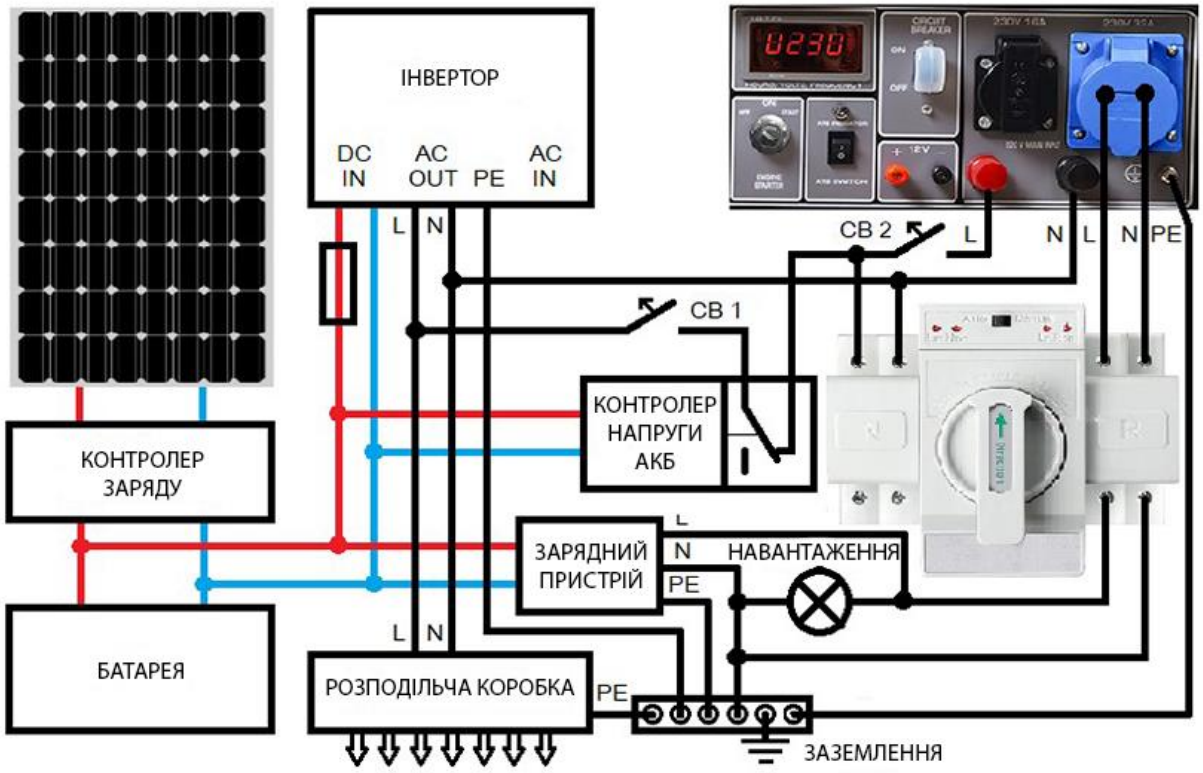


Рисунок 5.1 – Принципова схема підключення обладнання

Рекомендовані параметри заземлення для установки. У разі 1 кліматичної зони та верхньому шару ґрунту - чорнозему, а нижньому - глина. Становить 6

вертикальних заземлювачів. Діаметром 5см, довжиною 3 метри. Горизонтальний заземлювач встановлюється на глибині 1 метр, ширина полозки 5см. Довжина з'єднань 2 метри. У такому випадку загальний супротив становитиме 3,622 Ом.

Забороняється проводити електро-технічне обслуговування під час несприятливих погодних умов, таких як снігопад, дощ, град та інші види опадів.

У разі виконання робіт з електроустаткування, необхідно накривати сонячну батарею непрозорим листом, щоб «вимкнути» генерацію енергії від сонця. Завжди блокуйте та знеструмлюйте джерела постійного та змінного струму. Перевіряйте схеми, щоб переконатися, що вони знеструмлені, перш ніж працювати з ними.

Використовуйте струмовий затискач або вольтметр, щоб перевірити наявність небезпечної енергії перед роботою з фотоелектричними панелями. Будьте особливо обережні з інверторами, вони здатні утримувати заряд навіть при відключенні живлення.

Ніколи не від'єднуйте фотоелектричні роз'єми, затискачі, клемники або інше обладнання, яке знаходиться під навантаженням. Носити виключно спец одяг, який відповідає нормам електробезпеки.

Кабельні лінії які заводяться до шаф, прокладаються у металевому або пластиковому рукаві, з використання сигнальної стрічки. Та підводиться до шафи у металевій трубі. У разі використання броньованого кабелю, рукав можна не використовувати. Вся кабельна продукція повинна мати ізоляцію не підтримуючу горіння. На шафу з інвертором передбачене встановлення попереджувальної таблички. Біля вхідної клеми обов'язково встановлюється автоматичний вимикач, для захисту від струмів короткого замикання. Робочою напругою 220В та струмом спрацювання від короткого замкнення 2А. Наслідки недотримання цих вимог зображені на рисунку 5.2.

Рисунок 5.2 – Наслідкі не дотримань вимог з безпеки

Встановлювання вимикача після інвертора не є доцільним, бо він має власний вбудований у корпус автоматичний вимикач.

Забороняється використовувати редуктори відкритого типу, без захисного кожуха. Небезпечні частини рухомої конструкції виділяються окремим кольором. А на найвищу балку встановлюється металевий гачок, для закріплення захисного поясу. Не допускається обслуговування без повідомлення оператора, який повинен обов'язково вимкнути автоматичний режим роботи установки. У разі коли роботу виконує черговий інженер, установку слід зупинити завчасно.

Перед використанням підйомного приладдя перевірте його на цілісність. Якщо знайшли дефекти позначте стрічкою або етикеткою «Не використовувати», «Технічно не справно». Використовуйте драбини зі скловолокна чи інших діелектричних матеріалів, біля джерела живлення. Металеві або алюмінєві сходи є небезпечними поблизу ліній електропередачі або електричних робіт.

Встановлювати сходи дозволено лише на сухій рівній землі та на безпечній відстані від ліній електропередачи. Закріплюйте сходи на землі або на верхній

частині металоконструкції трекера. Беріться лише за горизонтальні сходинки, а не за вертикальні рейки та дотримуйтесь 3 точок хвату.

Ніколи не піднімайте сонячну панель або інше обладнання під час підйому по драбині. Використовуйте лебідку або систему підйому.

Перед початком роботи визначайте усі потенційні небезпеки, спіткнутися та впасти, це дозволить мінімізувати ризик отримання травм. Тримайте робоче місце в чистоті. Постійно переконуйтесь, що на робочій поверхні не містяться залишки мастила, льоду, води або інших речовин.

Для транспортування та заміни сонячних елементів у разі потреби використовуються візки, вилочні навантажувачі або інші методи, які не виконуються вручну.

Використовуйте рукавички, щоб захистити руки та покращити зчеплення. Завжди тримайте сонячні елементи двома руками, та використовуй плавні та рівні рухи. Тримайте вантаж як умога ближче. Не скручуйте тіло під час переносу вантажу, зробіть крок в одну чи іншу сторону, щоб повернутись. Не дозволяється замінювати розбиті елементи без захисних окулярів, рукавиць, та закритого одягу. Під час роботи у сонячну погоду, використовуйте головний убір, та сонцезахисні окуляри.

5.3 Пожежна профілактика

Для оцінки вибухопожежної та пожежної небезпеки використовується нормативний документ ДСТУ Б В.1.1-36:2016. Згідно якого, установка належить до категорії «Д» відповідно до таблиці 6, цього документа. В ній відсутні вибухонебезпечні речовини та мінімізована кількість речовин які підтримують горіння. Основним джерелом пожежі є вихід з ладу обладнання або перегрів сонячних фотоелектричних модулів. У сонячних модулях присутня не велика кількість полімерних капсул що оточують фотоелементи, полімерні листи основи, пластикові з’єднання коробки на задній панелі.

Для забезпечення відмінної пожежної безпеки та остаточного запобігання займання установки с подальшим розповсюдженням вогню через суху траву або інші горючі матеріали, розробленні профілактичні міри.

Не допускається накопичення сміття, заростів чи інших видів пожежонебезпечних об’єктів на майданчику знаходження установки. Зона у якій монтується пожежний щит, повинна мати додаткове освітлення, для швидкого знаходження при недостатній освітленості.

Необхідно забезпечити наявність заправлених вогнегасників, наповнених контейнерів з піском. Для тушіння установки використовувати порошкові вогнегасники, які встановлюються у пожарному щиті та слід застосувати для гасіння твердих, рідких, газоподібних речовин, електроустановок під напругою до 1000В. Кількість вогнегасників ВП-5 (Рис 5.2) повинна відповідати кількості установок.

Рисунок 5.2 – Вогнегасник ВП-5

Для запобігання розповсюдження вогню по природному середовищу використовується протипожежне водопостачання (Рис5.3) низького тиску. Джерелом води можу слугувати озера або річки, водопроводи, протипожежні резервуари.

Рисунок 5.3 – Протипожежне водопостачання.

Для оповіщення про появу пожежі використовуються сигналізації. Вони здатні привернути увагу людей, які працюють на підприємстві та повідомити пожежну охорону про спрацьовування. Для об’єкту на відкритій місцевості застосуються датчика вогню (Рис 5.4).

Рисунок 5.4 – Інфрачервоний датчик вогню

Датчик встановлюється на металевій опорі висотою 20 метрів, та дозволяє контролювати наявність полум’я на площинах до 200 квадратних метрів. Час спрацювання такого датчику становить 5 секунд. Біля пожежних щитів також

встановлюються пороговий ручний сповіщувач, для виклику пожежників самостійно.

Рисунок 5.5 – Ручний сповіщувач

ВИСНОВКИ

Відповідно до мети та об’єкту, була підібрана оптимальна конструкція та спосіб керування установкою. На основі чого, були створенні вимоги до об’єкта, його вхідні та вихідні параметри. Проробленні елементи підвищення надійності та безпеки експлуатації. Використання сучасного обладнання зменшує собівартість виготовлення та модернізації існуючих систем.

Спираючись на обрані апаратні рішення, створене програмне забезпечення, яке дозволяє повноцінно реалізувати всі функції керування. А саме: автоматичне позиціонування за координатами сонця, автоматичне розпізнавання хмарності та снігопадів, ручне керування, збір та відображення архівної інформації, дистанційне керування об’єктом через віддалений робочий стіл, веб сайт, мобільний додаток.

Проведений аналіз економічної доцільності, що є ключовим фактором впровадження будь якої системи автоматизації. Створенні та прораховані максимально безпечні умови праці. Що дозволить зберегти здоров’я працівників та збільшити привабливість робочого місця.

Розроблена система може легко піддаватись модернізації та масштабуватись. А реалізовані рішення та собівартість створення або покращення існуючих комплексів, дозволяють створити непоганий конкурентний вплив. Впровадження системи значно збільшує прибутковість фотоелектричних установок та примножує їх привабливість, що в свою чергу дозволить розповсюдити продукт на більшу аудиторію.

Мета та ціль дипломної роботи є повністю виконанні, відповідно до вимог та узгоджені з усіма консультантами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Методичні рекомендації для студентів бакалаврів спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» Ткачов В.В., Бубліков А.В., Цвіркун Л.І., Проценко С.М.., Бойко О.О., Славинський Д.В., .– Д.: «НГУ», 2016. – 27 с.
2. Небесна механіка Ю.В. Олександров – Харків «ХНУ», 2006. – 195 с.
3. Авіаційна астрономія М.А. Чорний – Москва «Транспорт» - 1978. – 208с.
4. Електронний ресурс: https://ecosolution.com.ua/catalog/detail/solnechnaya-panel-c-t-solar-320-vt-mono/
5. Електронний ресурс: https://darxton.ru/catalog_item/reduktor-chervyachnyy-rv40-86-50-1-dlya-shagovogo-dvigatelya/
6. Електронний ресурс: https://grizlicnc.com.ua/tovar-nema344-98mm-d14mm
7. Електронний ресурс: https://gosolar.com.ua/p1460331762-axioma-energy-ispwm.html?source=merchant_center&gclid=CjwKCAjwj42UBhAAEiwACIhADlMcex2PYP2WCSTPGgz3AKFJmpa2JvmGrKofrLoUihpIbTkcBC6o_hoCOBwQAvD_BwE
8. Ресурс «Економічна Правда» - https://www.epravda.com.ua
9. Ресурс «FORBES» - https://forbes.ua/ru/company/zelenaya-energetika-v-ukraine-kak-istoriya-mezhdunarodnogo-uspekha-obernulas-ocherednym-provalom-26052021-1676
10. Ресурс «Нова газета» - https://biz.nv.ua/markets/vozobnovlyaemaya-energetika-v-ukraine-itogi-goda-i-prognozy-2022-vie-50203541.html
11. Правила виконання робочої документації автоматизації технологічних процесів. ДСТУ Б А.2.4-3:2009. Київ, 2009 – 54с.
12. Визначення категорій приміщень, будинків та зовнішніх установок за вибохопожежною та пожежною небезпекою. ДСТУ Б В.1.1-36:2016 Київ, 2016. – 31с.
13. Електронний ресурс: https://dnaop.com
14. Прикладна механіка В. Булгаков, О. Черниш, В. Яременко – 2018. - 612с.

15. Електронний ресурс: https://www.bre.co.uk/page.jsp?id=3211
16. Безпека й аварійне реагування на сонячних електростанціях Casey C. Grant, P.E. USA – 2013 – 99с
17. Електронний ресурс: https://www.dovepress.com/emerging-osh-issues-in-installation-and-maintenance-of-floating-solar--peer-reviewed-fulltext-article-RMHP
18. Електронний ресурс: https://www.graphicproducts.com/infographics/solar-power-hazards-and-safety/
19. Електронний ресурс: https://ru.wikipedia.org
20. Електронний ресурс: https://www.highexpert.ru/methods/wind_advertise-.html

ДОДАТОК А

Відомість матеріалів кваліфікаційної роботи

№ рядка	Формат	Позначення	Найменування	Кількість аркушів	Шифр документу	Примітка
1			**Документація**			
2						
3	А4	КФІВС.КВР.151.18.15.ПЗ	Пояснювальна записка	127	ПЗ	
4						
5			**Графічні матеріали**			
6						
7	А2	КФІВС.КВР.151.18.15.Е2	Функціональна схема			
8			автоматизації	1	Е2	
9						
10	А2	КФІВС.КВР.151.18.15.Е3	Схема електрична			
11			принципова	1	Е3	
12						
13	А2	КФІВС.КВР.151.18.15.Е4	Схема електрична	1	Е4	
14			з’єднань			
15						
16	А4	КФІВС.КВР.151.18.15.ПЕ	Перелік елементів	2	ПЕ	
17						
18	А4	КФІВС.КВР.151.18.15.Д	Презентація		Д	
19						
20		КФІВС.КВР.151.18.15.ВДЕ	Носій інформації	1	ВДЕ	
21						
22						
23						
24						
25						
26						
27						
28						
29						

			Підп.	Дата	**КФІВС.КВР.151.18.15.ТП**		
Зм.	Арк.	№ докум.					
Розробив		Рульов		03.06	Автоматизація процесів керування сонячною фотоелектричною установкою Відомість проекту	Літ.	Аркуш 1 / Аркушів 1
П. конс.		Зибалов					
Н. контр.		Славінський				Національний ТУ «Дніпровська політехніка», ЕФ, 151-18-1	

ДОДАТОК Б

Перелік елементів схеми електричної принципової

Позначення	Найменування	Кількість	Примітка
DA1	L7805ABV	1	
DA2,DA4	LM358	2	
DA3	ACS714-50A	1	
DA5	MCP1525	1	
DD1	MCP2210	1	
DD2	MCP3302	1	
DD3	TB6600HG	1	
FU1	32S-060H	1	Запобіжник 6А
L1	AMI321611-R22M-NP 20% 150nH	1	
L2	AMI321611-R22M-NP 20% 10uH	1	
V1,V2	KLS9-L-5013GC	2	Світлодіод зелений
ZQ1	KX-49 12 MHz	1	
R1	Panasonic ERJ6ENF 1K Om	1	
R2,R14	Panasonic ERJ6ENF 100K Om	2	
R3,R4	Panasonic ERJ6ENF 560 Om	2	
R5,R6	Panasonic ERJ6ENF 4.7K Om	2	
R7	Panasonic ERJ6ENF 1K Om	1	
R8	Panasonic ERJ6ENF 1.27K Om	1	
R9	Panasonic ERJ6ENF 10K Om	1	
R10,R11	Panasonic ERJ6ENF 0.33 Om	2	
R12	Panasonic ERJ6ENF 47K Om	1	
R13	Panasonic ERJ6ENF 4.7K Om	1	
C1,C2	Hitano C0805N180J500NT 18pF	2	
C3,C4,C6,C8, C9,C11,C16, C17,C19,C20	Hitano C0805N101J500NT 100pF	10	
C5	Hitano C0805N225J250N3 2.2uF	1	

			Підп.	Дата	**КФІВС.КВР.151.18.15.ПЕ3**		
Зм.	Арк.	№ докум.					
Розробив		Рульов		24.05	Автоматизація процесів керування сонячною фотоелектричною установкою Перелік елементів	Літ. / Аркуш 1 / Аркушів 2	
П. конс.		Зибалов					
Консультант		Проценко				Національний ТУ «Дніпровська політехніка», ЕФ, 151-18-1	
Н. контр.		Славінський					

Позначення	Найменування	Кількість	Примітка
C7	Nichicon 63V PM 12.5x35.5 470uF	1	Електроліт
C10	Hitano C0805B102J500NT 1nF	1	
C12,C13,C15	Hitano C0805N105K100N3 1uF	3	
C14	Panasonic 25V FC 12.5x20mm 1000uF	1	Електроліт
C18	Hitano C0805B105K160N3 10uF	1	
X1	USB-B KLS1-151-W	1	
X2	MF-2MA	1	
X3	KLS2-126-5.00-02P-4Cx3	1	
X4	MKDSP-10N/2	1	
X5	MF-4MA	1	

			Підп.	Дата	**КФІВС.КВР.151.18.15.ПЕ3**	
Зм.	Арк.	№ докум.				
Розробив		Рульов		24.05	Автоматизація процесів керування сонячною фотоелектричною установкою Перелік елементів	Літ. / Аркуш 2 / Аркушів 2
П. конс.		Зибалов				
Консультант		Проценко				Національний ТУ «Дніпровська політехніка», ЕФ, 151-18-1
Н. контр.		Славінський				

ДОДАТОК В

Код програмного забезпечення

Форма MainForm

```
////////////////////////////////////////////////////////////////////////////////
using Data_writer.Properties;
using MaterialSkin;
using MaterialSkin.Controls;
using mcp2210_dll_m;
using Newtonsoft.Json;
using MQTTnet;
using MQTTnet.Client.Connecting;
using MQTTnet.Client.Disconnecting;
using MQTTnet.Client.Options;
using MQTTnet.Extensions.ManagedClient;
using System;
using System.Threading;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Media;
using System.Net;
using System.Text;
using System.Windows.Forms;
using System.Threading.Tasks;

namespace Data_writer
{
    public partial class FormMain : MaterialForm
    {
        byte led_blink_set = 0; //Режим работы светодиодов в мигании.
        byte numb_item = 0;
        byte C = 0, state = 0;
        bool control_server = false; //Флаг управления от сервера.
        bool activ_report_solar;
        bool dark_theme;
        bool power_low;
        bool activ_log = Convert.ToBoolean(Settings.Default["activ_log"]);
        int GET_interval = Convert.ToInt32(Settings.Default["polling_time"]);
//Интервал GET запросов, в секундах.
        int time_out = Convert.ToInt32(Settings.Default["time_out"]);
        int time_out_count;
        string weather;
        string response_server = "";
        string url_get =
"http://smartdnepr.biz.ua/get.php?p=OFFLINEVOLT&i=OFFLINECURRENT";
        double construction_angle = 20.00;
        double high_solar, azimuth_solar;
        // Переменные для АЦП
        double[] buffer_volteg = new double[5];
        double[] buffer_curent = new double[5];
        double[] buffer_angle = new double[5];
        double volteg;
        double curent;
        double power;
        double angle;
        //MQTT
```

```
        int portMQTT =
Convert.ToInt32(Settings.Default["mqtt_port"].ToString());
        string adress_brokerMQTT = Settings.Default["mqtt_ip"].ToString();
        string loginMQTT = Settings.Default["mqtt_login"].ToString();
        string passwordMQTT = Settings.Default["mqtt_password"].ToString();
        bool use_local_broker =
Convert.ToBoolean(Settings.Default["mqtt_broker"]);
        string[] android_data = new string[] { "false", "false", "false",
"false", "35" }; //0-load1 1-load2 2-RemoteControlLoad 3-AutomaticControl 4-
TiltAngle
        //
        public string DatabaseLink; // Не безопасный путь базы данных.
        FormInfoProgram FormInfo;
        FormSetting SettForm;
        GrafForm GrafShow;
        SqlConnection sqlConnection;
        FormUpdt FormUpdt;
        Thread GetThread;
        Thread MqttThread;

        double[] buffer_adc = new double[3];
        uint usb_connection_attempts = 10;
        const ushort DEFAULT_PID_MODUL_BOX = 0xde;
        const ushort DEFAULT_PID_MODUL_SOLAR = 0xde;
        IntPtr modul_box;
        IntPtr modul_solar;

        FileManager File = new FileManager();
        HardwareMCP2210 HardwareMCP2210 = new HardwareMCP2210();
        Net Net = new Net();
        NetMQTT netMQTT = new NetMQTT();

        SoundPlayer info = new SoundPlayer(Resources.info);
        Image image_solat_panel = Properties.Resources.SolarPanel;

        public FormMain()
        {
            InitializeComponent();

            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.EnforceBackcolorOnAllComponents = true;
            materialSkinManager.AddFormToManage(this);

            bool dark_theme = Convert.ToBoolean(Settings.Default["dark_theme"]);
            if (dark_theme)
            {
                materialSkinManager.Theme = MaterialSkinManager.Themes.DARK;
                materialSkinManager.ColorScheme = new ColorScheme(
                        materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal500 : Primary.Indigo500,
                        materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal700 : Primary.Indigo700,
                        materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal200 : Primary.Indigo100,
                        Accent.Pink200,
                        TextShade.WHITE);
            }
            else
            {
                materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;
                materialSkinManager.ColorScheme = new
ColorScheme(Primary.Indigo500, Primary.Indigo700, Primary.Indigo100,
Accent.Pink200, TextShade.WHITE);
            }
```

```
            SetGrafSolid(dark_theme); // вызов функции базовой настройки
анимированых шкал
            File.FileMain();
            File.ClearUpdate();

            Program.FM = this;
            GetThread = new Thread(GetCall);
            MqttThread = new Thread(MqttCall);

            File.FileWriteDataLog("Program launch", "info", activ_log);

            activ_report_solar =
Convert.ToBoolean(Settings.Default["activ_report_solar"]);
            if (activ_report_solar == true)
            {
                timer_write_text_file.Interval = (1000 *
Convert.ToInt16(Settings.Default["report_writing_frequency"]));
                timer_write_text_file.Enabled = true;
            }
            else
            {
                timer_write_text_file.Enabled = false;
            }

            if (use_local_broker)
            {
                //netMQTT.StartBroker();
            }

            netMQTT.InitializationMQTT(adress_brokerMQTT, portMQTT);

            MqttThread.Start();
            GetThread.Start();
        }

        public void SetGrafSolid(bool dark_them) //настройка шкал
        {
            solidGaugevolteg_solar_panel.To = 42;

            solidGaugecurent.To = 45;
            solidGaugepower.To = 1400;
            solidGaugeAngle.To = 160;
            solidGaugeAngle.From = 20;

            solidGaugevolteg_solar_panel.ToColor =
System.Windows.Media.Color.FromRgb(0, 150, 40);
            solidGaugeAngle.ToColor = System.Windows.Media.Color.FromRgb(255,
140, 0);
            solidGaugeAngle.FromColor = System.Windows.Media.Color.FromRgb(255,
140, 0);

            solidGaugepower.ToColor = System.Windows.Media.Color.FromRgb(0, 150,
40);
            solidGaugecurent.ToColor = System.Windows.Media.Color.FromRgb(205,
92, 92);
            solidGaugecurent.FromColor = System.Windows.Media.Color.FromRgb(0,
150, 40);

            if (dark_them)
            {
                solidGaugevolteg_solar_panel.Base.Foreground =
System.Windows.Media.Brushes.WhiteSmoke;
```

```
                solidGaugepower.Base.Foreground =
System.Windows.Media.Brushes.WhiteSmoke;
                solidGaugecurent.Base.Foreground =
System.Windows.Media.Brushes.WhiteSmoke;
                solidGaugeAngle.Base.Foreground =
System.Windows.Media.Brushes.WhiteSmoke;
            }
            else
            {
                solidGaugevolteg_solar_panel.Base.Foreground =
System.Windows.Media.Brushes.Black;
                solidGaugepower.Base.Foreground =
System.Windows.Media.Brushes.Black;
                solidGaugecurent.Base.Foreground =
System.Windows.Media.Brushes.Black;
                solidGaugeAngle.Base.Foreground =
System.Windows.Media.Brushes.Black;
            }
        }
        void ColorReset() //Сброс цвета кнопок.
        {
            pictureLamp1.Image = Properties.Resources.выключить_свет;
            pictureLamp2.Image = Properties.Resources.выключить_свет;
            Button_on_off_led_1.Text = "ON";
            Button_on_off_led_2.Text = "ON";
        }
        private void GetCall()
        {
            Thread.Sleep(100);
            while (true)
            {
                if (Convert.ToInt16(DateTime.Now.Hour) >= 8 &&
Convert.ToInt16(DateTime.Now.Hour) <= 15)
                {
                    Invoke((MethodInvoker)(() =>
                    {
                        pictureup.Visible = true;
                        TimerLinkImage.Enabled = true;
                        picturedown.Visible = true;
                    }));
                    if (Net.Transmitter(ref response_server, url_get))
                    {
                        if (server_stat_label.Text != "ONLINE")
                        {
                            File.FileWriteDataLog("Server connection
established", "info", activ_log);
                        }
                        Invoke((MethodInvoker)(() =>
                        {
                            pictureup.Visible = false;
                            label_time_out.Visible = false;
                            GET_info.Text = response_server.ToUpper();
                            server_stat_label.Text = "ONLINE";
                            TimerLinkImage.Enabled = true;
                            pictureServer.Image =
Properties.Resources.двойная_стрелка_вправо;
                        }));

                        Thread.Sleep(GET_interval * 1000);
                    }
                    else
                    {
                        if (server_stat_label.Text != "OFFLINE")
                        {
```

```
                            File.FileWriteDataLog("No connection to server",
"error", activ_log);
                        }
                        Invoke((MethodInvoker)(() =>
                        {
                            pictureup.Visible = false;
                            picturerimeout.Visible = true;
                            server_stat_label.Text = "OFFLINE";
                            server_stat_label.ForeColor =
System.Drawing.Color.Red;
                            time_out_count = time_out;

                            MaterialSwitchServerOnOff.Enabled = false;
                            MaterialSwitchServerOnOff.Checked = false;
                            pictureServer.Image =
Properties.Resources.высокий_приоритет;
                        }));
                        Thread.Sleep(time_out * 1000);
                    }
                }
                else
                {
                    if (server_stat_label.ForeColor != Color.Yellow)
                    {
                        File.FileWriteDataLog("The server is disabled at the end
of the working day", "info", activ_log);

                        Invoke((MethodInvoker)(() =>
                        {
                            pictureup.Visible = false;
                            server_stat_label.Text = "OFFLINE";
                            server_stat_label.ForeColor = Color.Yellow;
                            MaterialSwitchServerOnOff.Enabled = false;
                            MaterialSwitchServerOnOff.Checked = false;
                            pictureServer.Image =
Properties.Resources.высокий_приоритет;
                        }));
                    }
                    Thread.Sleep(60 * 1000);
                }
            }
        }
        private void MqttCall()
        {
            Thread.Sleep(100);
            Random rnd = new Random();
            while (true)
            {
                double[] data = { rnd.Next(0, 25), rnd.Next(0, 25), rnd.Next(0,
25), rnd.Next(0, 25), rnd.Next(0, 25), rnd.Next(0, 25), rnd.Next(0, 25), };
                if (netMQTT.PublishData(data, android_data))
                {
                    Invoke((MethodInvoker)(() =>
                    {
                        MqttStatusPictureBox.Image =
Properties.Resources.mqtt_в_сети_96;
                    }));
                    Thread.Sleep(5000);
                }
                else
                {
                    Invoke((MethodInvoker)(() =>
                    {
```

```
                        MqttStatusPictureBox.Image =
Properties.Resources.mqtt_не_в_сети_96;
                    }));
                    Thread.Sleep(10000);
                }
            }
        }
        private void StepMotorRun(double sen_angle)
        {
            string ret_info;
            ret_info = HardwareMCP2210.StepMotorControl(sen_angle, angle,
modul_box);

            if (ret_info.Contains("error"))
            {
                File.FileWriteDataLog(ret_info, "error", activ_log);
            }
            else
            {
                File.FileWriteDataLog(ret_info, "info", activ_log);
            };
        }
        void SolarCoordinates()
        {
            const double LONGITUDE = 35.062396061280175;
            const double LATITUDE = 48.45534994945111;
            double[,] sun_declination_table = {
    {-23.067, -22.983, -22.9, -22.8, -22.7, -22.6, -22.467, -22.35, -22.217, -
22.083, -21.933, -21.783, -21.617, -21.45, -21.267, 21.1, -20.9, -20.7, -20.5, -
20.18, -20.05, -19.867, -19.633, -19.4, -19.167, -18.917, -18.667, -18.417, -
18.15, -17.883, -17.617 },
    {-17.333, -17.05, -16.767, -16.467, -16.167, -15.867, -15.567, -15.25, -
14.933, -14.617, -14.3, -13.967, -13.633, -13.3, -12.967, -12.617, -12.267, -
11.917, -11.567, -11.217, -10.867, -10.5, -10.133, -9.767, -9.4, -9.033, -8.65,
-8.283, -8.05, double.NaN, double.NaN},
    {-7.817, -7.433, -7.05, -6.667, -6.283, -5.9, -5.5, -5.117, -4.733, -4.333,
-3.95, -3.55, -3.167, -2.767, -2.367, -1.983, -1.583, -1.183, -0.8, -0.4, 0.00,
0.4, 0.783, 1.183, 1.583, 1.967, 2.367, 2.75, 3.15, 3.533, 3.917 },
    {4.3, 4.07, 5.083, 5.467, 5.85, 6.217, 6.6, 6.983, 7.35, 7.717, 8.117,
8.467, 8.833, 9.183, 9.55, 9.9, 10.267, 10.616, 10.967, 11.317, 11.65, 12.00,
12.333, 12.667, 13.00, 13.317, 13.633, 13.967, 14.267, 14.583, double.NaN },
    {14.9, 15.2, 15.5, 15.783, 16.083, 16.367, 16.65, 16.917, 17.2, 17.45,
17.717, 17.983, 18.233, 18.483, 18.716, 18.967, 19.183, 19.417, 19.633, 19.85,
20.067, 20.267, 20.467, 20.65, 20.833, 21.017, 21.2, 21.367, 21.517, 21.683,
21.833 },
    {21.967, 22.1, 22.233, 22.367, 22.483, 22.583, 22.7, 22.783, 22.53, 22.883,
23.033, 23.117, 23.183, 23.233, 23.283, 23.333, 23.367, 23.4, 23.417, 23.433,
23.433, 23.433, 23.433, 23.417, 23.4, 23.383, 23.35, 23.317, 23.267, 23.217,
double.NaN },
    {23.15, 23.083, 23.017, 22.93, 22.85, 22.75, 22.65, 22.55, 22.433, 22.317,
22.183, 22.067, 21.917, 21.767, 21.617, 21.467, 21.3, 21.133, 20.97, 20.783,
20.6, 20.4, 20.2, 20.00, 19.783, 19.567, 19.35, 19.133, 18.9, 18.667, 18.417 },
    {18.167, 17.917, 17.667, 17.4, 17.133, 16.867, 16.6, 16.316, 16.033, 15.75,
15.45, 15.167, 14.867, 14.55, 14.25, 13.933, 13.617, 13.3, 12.983, 12.65,
12.317, 11.983, 11.65, 11.317, 10.967, 10.633, 10.283, 9.933, 9.583, 9.217,
8.867 },
    {8.5, 8.15, 7.783, 7.417, 7.05, 6.667, 6.3, 5.933, 5.55, 5.167, 4.8, 4.417,
4.033, 3.65, 3.267, 2.883, 2.5, 2.1, 1.717, 1.333 ,0.95, 0.55, 0.167 , -0.233, -
0.617, -1.00, -1.4, -1.783, -2.167, -2.567, double.NaN },
    {-2.95, -3.333, -3.733, -4.117, -4.5, -4.883, -5.267, -5.65, -6.033, -6.417,
-6.8, -7.167, -7.533, -7.917, -8.3, -8.667, -9.033, -9.4, -9.75, -10.117, -
10.483, -10.833, -11.2, -11.55, -11.9, -12.233, -12.583, -12.917, -13.25, -
13.583, -13.917 },
```

```
    {-14.233, -14.567, -14.883, -15.017, -15.5, -15.8, -16.1, -16.4, -16.683, -
16.967, -17.083, -17.533, -17.8, -18.067, -18.333, -18.583, -18.833, -19.083, -
19.317, -19.55, -19.783, -20.00, -20.217, -20.433, -20.633, -20.833, -21.017, -
21.2, -21.383, -21.55, double.NaN },
    {-21.717, -21.867, -22.017, -22.167, -22.3, -22.417, -22.533, -22.65, -
22.767, -22.867, -22.95, -23.033, -23.117, -23.183, -23.233, -23.283, -23.333, -
23.367, -23.4, -23.417, -23.433, -23.433, -23.433, -23.433, -23.417, -23.383, -
23.35, -23.317, -23.267, -23.2, -23.133 }
};

            double EOT_time, EOT_variable, true_solar_time, UTC_time,
astronomical_time;
            int day_year;
            DateTime current_time;

            current_time = DateTime.UtcNow;
            day_year = current_time.DayOfYear;
            ////////////////////////////////////////////////
            UTC_time = (current_time.Hour * 60);
            UTC_time += (current_time.Minute);
            UTC_time += ((double)current_time.Second / 60);
            ////////////////////////////////////////////////
            EOT_variable = ((double)(360 * (day_year - 81)) / 365);

            //Нахождение уравнение времени EOT.
            EOT_time = 9.87 * Math.Sin((2 * EOT_variable) * Math.PI / 180) -
7.53 * Math.Cos(EOT_variable * Math.PI / 180) - 1.5 * Math.Sin(EOT_variable *
Math.PI / 180); //Результат минуты.

            //Расщёт истинного астрономического времени.
            astronomical_time = UTC_time + (LONGITUDE * 4) + EOT_time;

            //Расщёт истинного времение светила.
            true_solar_time = astronomical_time - (12 * 60);

            //Расщёт высоты солнца.
            high_solar =
Math.Asin(Math.Sin(sun_declination_table[(current_time.Month - 1),
(current_time.Day - 1)] * Math.PI / 180) * Math.Sin(LATITUDE * Math.PI / 180) +
Math.Cos(sun_declination_table[(current_time.Month - 1), (current_time.Day - 1)]
* Math.PI / 180) * Math.Cos(LATITUDE * Math.PI / 180) *
Math.Cos((true_solar_time / 4) * Math.PI / 180));
            high_solar = high_solar * 180 / Math.PI;

            MaterialLabelHighSolar.Text = Math.Round(high_solar, 2).ToString() +
"°";
            //Расщёт азимута солнца.
            azimuth_solar =
Math.Atan2(Math.Cos(sun_declination_table[(current_time.Month - 1),
(current_time.Day - 1)] * Math.PI / 180) * Math.Sin((true_solar_time / 4) *
Math.PI / 180), Math.Cos(sun_declination_table[(current_time.Month - 1),
(current_time.Day - 1)] * Math.PI / 180) * Math.Sin(LATITUDE * Math.PI / 180) *
Math.Cos((true_solar_time / 4) * Math.PI / 180) -
Math.Sin(sun_declination_table[(current_time.Month - 1), (current_time.Day - 1)]
* Math.PI / 180) * Math.Cos(LATITUDE * Math.PI / 180));
            azimuth_solar = (azimuth_solar * 180 / Math.PI) + 180;

            MaterialLabelAzimus.Text = Math.Round(azimuth_solar, 2).ToString() +
"°";
        }
        void AutoControl()
        {
            if (high_solar >= 0)
            {
```

```
                if (azimuth_solar >= 90 && azimuth_solar <= 110)
                {
                    StepMotorRun(20);
                }
                else if (azimuth_solar >= 160 && azimuth_solar <= 180)
                {
                    StepMotorRun(160);
                }
                else if (azimuth_solar > 110 && azimuth_solar < 160)
                {
                    if ((weather == "Cloud" || weather == "Fog" || weather ==
"Partly Cloudy") && power_low == true)
                    {
                        StepMotorRun(90);
                    }
                    else if (weather == "Snow" && power_low == true)
                    {
                        if ((angle - 90) > 0)
                        {
                            StepMotorRun(160);
                        }
                        else
                        {
                            StepMotorRun(20);
                        }
                    }
                    else
                    {
                        StepMotorRun(azimuth_solar - 90);
                    }
                }
            }
            else
            {
                StepMotorRun(20);
            }
        }


//////////////////////////////////////Кнопки/////////////////////////////////////
//////////
        private void Connect_Click(object sender, EventArgs e)
        {
            if (button_conect.Text == "Підключитися")
            {
                string answer =
HardwareMCP2210.UsbConnect(DEFAULT_PID_MODUL_BOX, ref modul_box);
                if (answer.Contains("Module not found"))
                {
                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Модуль не знайдено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);
                    ////////////////////////////////////////////////////
                    //pictureUSB.Image = Properties.Resources.usb_отключен;
                    //ColorReset();
                    ////////////////////////////////////////////////////
                }
                else if (answer.Contains("Module already connected"))
                {
                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Модуль вже підключено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);
```

```
                }
                else
                {
                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Модуль підключено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);

                    File.FileWriteDataLog(answer, "info", activ_log);

                    button_conect.Text = "Відключитися";
                    TimerUsbProtect.Enabled = true;
                    Button_on_off_led_1.Enabled = true;
                    Button_on_off_led_2.Enabled = true;
                    MaterialSwitchServerOnOff.Checked = true;
                    pictureUSB.Image = Properties.Resources.usb_подключен;
                }
            }
            else if (button_conect.Text != "Підключитися")
            {
                if (HardwareMCP2210.UsbDisconect(modul_box))
                {
                    button_conect.Text = "Підключитися";
                    pictureUSB.Image = Properties.Resources.usb_отключен;

                    Button_on_off_led_1.Enabled = false;
                    Button_on_off_led_2.Enabled = false;
                    timer_blink.Enabled = false;
                    TimerADC.Enabled = false;
                    ColorReset();
                    File.FileWriteDataLog("Module disabled by user", "info",
activ_log);
                    HardwareMCP2210.ControlPorts(0, 0, modul_box);

                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Модуль вимкнено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);
                }
            }
        }
        private void Button_on_off_led_1_Click(object sender, EventArgs e)
        {
            if (control_server == false && Button_on_off_led_1.Text == "ON")
            {
                HardwareMCP2210.ControlPorts(1, 100, modul_box);
                Button_on_off_led_1.Text = "OFF";
                pictureLamp1.Image = Properties.Resources.включить_свет;
                android_data[0] = "true";
            }
            else if (control_server == false && Button_on_off_led_1.Text ==
"OFF")
            {
                HardwareMCP2210.ControlPorts(0, 100, modul_box);
                Button_on_off_led_1.Text = "ON";
                pictureLamp1.Image = Properties.Resources.выключить_свет;
                android_data[0] = "false";
            }
            else
            {
                MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Отключите удалённое упраление.", "OK", true);
                info.Play();
                SnackBarMessage.Show(this);
```

```
            }
        }
        private void Button_on_off_led_2_Click(object sender, EventArgs e)
        {
            if (control_server == false && Button_on_off_led_2.Text == "ON")
            {
                HardwareMCP2210.ControlPorts(1, 101, modul_box);
                Button_on_off_led_2.Text = "OFF";
                pictureLamp2.Image = Properties.Resources.включить_свет;
                android_data[1] = "true";
            }
            else if (control_server == false && Button_on_off_led_2.Text ==
"OFF")
            {
                HardwareMCP2210.ControlPorts(0, 101, modul_box);
                Button_on_off_led_2.Text = "ON";
                pictureLamp2.Image = Properties.Resources.выключить_свет;
                android_data[1] = "false";
            }
            else
            {
                MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Отключите удалённое упраление.", "OK", true);
                info.Play();
                SnackBarMessage.Show(this);
            }
        }
        private void MaterialSwitchServerOnOff_Click(object sender, EventArgs e)
        {
            if (MaterialSwitchServerOnOff.Checked == true)
            {
                control_server = true;
                android_data[3] = "true";
                Control_Server_Setting();
            }
            else
            {
                control_server = false;
                android_data[0] = "false";
                Control_Server_Setting();
            }
        }
        private void MaterialSwitchAutoControl_CheckedChanged(object sender,
EventArgs e)
        {
            if (MaterialSwitchAutoControl.Checked)
            {
                MaterialSliderTungle.Enabled = false;
            }
            else
            {
                MaterialSliderTungle.Enabled = true;
            }
        }
        private void MaterialSliderTungle_Click(object sender, EventArgs e)
        {
            string ret_info;

            if (MaterialSliderTungle.Value >= 20)
            {
                if (angle > 19 && angle < 161)
                {
                    ret_info =
HardwareMCP2210.StepMotorControl(MaterialSliderTungle.Value, angle, modul_box);
```

```
                    if (ret_info.Contains("error"))
                    {
                        File.FileWriteDataLog(ret_info, "error", activ_log);
                    }
                    else
                    {
                        File.FileWriteDataLog(ret_info, "info", activ_log);
                    };

                    construction_angle = MaterialSliderTungle.Value;
                    PictureBoxSolarPanel.Invalidate();
                }
                else
                {
                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Немає зв'язку із датчиком! Переконайтеся, що модуль
підключено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);
                }
            }
            else
            {
                if (angle > 19 && angle < 161)
                {
                    ret_info =
HardwareMCP2210.StepMotorControl(MaterialSliderTungle.Value, angle, modul_box);

                    if (ret_info.Contains("error"))
                    {
                        File.FileWriteDataLog(ret_info, "error", activ_log);
                    }
                    else
                    {
                        File.FileWriteDataLog(ret_info, "info", activ_log);
                    };

                    MaterialSliderTungle.Value = 20;
                    construction_angle = 20;
                    PictureBoxSolarPanel.Invalidate();
                }
                else
                {
                    MaterialSnackBar SnackBarMessage = new
MaterialSnackBar("Немає зв'язку із датчиком! Переконайтеся, що модуль
підключено.", "OK", true);
                    info.Play();
                    SnackBarMessage.Show(this);
                }
            }
        }


//////////////////////////////////////Таймеры/////////////////////////////////////
///////////
        private void TimerUsbProtect_Tick(object sender, EventArgs e) // Таймер
отвечает за проверку подключения.
        {
            string answer = HardwareMCP2210.UsbCheckConnection(false);

            if (answer.Contains("All conect"))
            {
                if (control_server == true)
```

```
                {
                    MaterialSwitchServerOnOff.Checked = true;
                }

                MaterialSwitchServerOnOff.Enabled = true;
            }
            else
            {
                if (TimerUsbProtect.Interval != 5000)
                {
                    File.FileWriteDataLog(answer, "error", activ_log);
                }
                ushort pid =
Convert.ToUInt16(answer.Substring(answer.IndexOf("PID-") + 4));
                if (pid == DEFAULT_PID_MODUL_BOX)
                {
                    string answer_connect = HardwareMCP2210.UsbConnect(pid, ref
modul_box);
                    if (!answer_connect.Contains("Module connect"))
                    {
                        TimerUsbProtect.Interval = 5000;
                        if (usb_connection_attempts == 0)
                        {
                            usb_connection_attempts = 10;
                            HardwareMCP2210.UsbCheckConnection(true);
                            File.FileWriteDataLog((answer + "Failed to
reconnect"), "fatal", activ_log);

                            MaterialDialog materialDialog = new
MaterialDialog(this, "Потеря связи!", "Приложению не удалось установить связь с
модулем PID - " + pid.ToString() + ". Предпринято попыток " +
usb_connection_attempts.ToString() + ".", "OK", true, "Cancel");
                            materialDialog.ShowDialog(this);
                        }
                        usb_connection_attempts--;
                    }
                    else
                    {
                        TimerUsbProtect.Interval = 1000;
                        HardwareMCP2210.UsbCheckConnection(true);
                        File.FileWriteDataLog((answer_connect + " Connection
restored"), "info", activ_log);
                        if (MaterialSwitchServerOnOff.Checked)
                        {
                            Control_Server_Setting();
                        }
                    }
                }
                else if (pid == DEFAULT_PID_MODUL_SOLAR)
                {
                    string answer_connect = HardwareMCP2210.UsbConnect(pid, ref
modul_solar);
                    if (!answer_connect.Contains("Module connect"))
                    {
                        TimerUsbProtect.Interval = 5000;
                        if (usb_connection_attempts == 0)
                        {
                            usb_connection_attempts = 10;
                            HardwareMCP2210.UsbCheckConnection(true);
                            File.FileWriteDataLog((answer + "Failed to
reconnect"), "fatal", activ_log);

                            MaterialDialog materialDialog = new
MaterialDialog(this, "Потеря связи!", "Приложению не удалось установить связь с
```

```
модулем PID - " + pid.ToString() + ". Предпринято попыток " +
usb_connection_attempts.ToString() + ".", "OK", true, "Cancel");
                            materialDialog.ShowDialog(this);
                        }
                        usb_connection_attempts--;
                    }
                    else
                    {
                        TimerUsbProtect.Interval = 1000;
                        HardwareMCP2210.UsbCheckConnection(true);
                        File.FileWriteDataLog((answer_connect + " Connection
restored"), "info", activ_log);
                        if (MaterialSwitchServerOnOff.Checked)
                        {
                            Control_Server_Setting();
                        }
                    }
                }
            }
            if (state == 0)
            {
                HardwareMCP2210.ControlPorts(1, 107, modul_box);
                state = 1;
            }
            else
            {
                HardwareMCP2210.ControlPorts(0, 107, modul_box);
                state = 0;
            }
        }
        private void TimerLinkImage_Tick(object sender, EventArgs e)
        {
            picturedown.Visible = false;
            TimerLinkImage.Enabled = false;
        } //Для исчезновения иконок линка.
        private void TimerRealTime_Tick(object sender, EventArgs e) //Статичный
таймер с интервалом 1с.
        {
            double azimuth_temp;

            azimuth_temp = azimuth_solar;
            time_l.Text = DateTime.Now.ToShortTimeString();
            ap_db();

            if (picturerimeout.Visible == true && time_out_count > 0)
            {
                label_time_out.Visible = true;
                time_out_count -= 1;
                label_time_out.Text = Convert.ToString(time_out_count);
            }

            SolarCoordinates();

            if (MaterialSwitchAutoControl.Checked)
            {
                construction_angle = angle;
                MaterialSliderTungle.Value = Convert.ToInt32(angle);
                PictureBoxSolarPanel.Invalidate();

                if (azimuth_solar - 1 == azimuth_temp)
                {
                    AutoControl();
                }
            }
```

```
        }
        private void TimerADC_Tick(object sender, EventArgs e)
        {

            HardwareMCP2210.ADCDataReception(modul_box, ref buffer_adc);
            if (buffer_adc.Sum() == 0)
            {
                solidGaugevolteg_solar_panel.Value = 0;
                solidGaugecurent.Value = 0;
                solidGaugepower.Value = 0;
                solidGaugeAngle.Value = 0;

                curent = double.NaN;
                volteg = double.NaN;
                angle = double.NaN;
            }
            else
            {
                buffer_volteg[numb_item] = buffer_adc[0];
                buffer_curent[numb_item] = buffer_adc[1];
                buffer_angle[numb_item] = buffer_adc[2];

                numb_item++;
                if (numb_item > 4)
                {
                    curent = Math.Round((buffer_curent.Sum() / 5), 2);
                    volteg = Math.Round((buffer_volteg.Sum() / 5), 2);
                    angle = Math.Round((buffer_angle.Sum() / 5), 2);
                    power = Math.Round((curent * volteg), 1);

                    solidGaugevolteg_solar_panel.Value = volteg;
                    solidGaugecurent.Value = curent;
                    solidGaugepower.Value = power;
                    solidGaugeAngle.Value = angle;

                    numb_item = 0;
                    url_get = ("http://smartdnepr.biz.ua/get.php?p=" +
Convert.ToString(volteg) + "&i=" + Convert.ToString(curent));
                }
            }

        } //Таймер опроса АЦП.
        private void TimerGetWither_Tick(object sender, EventArgs e) //Таймер
опроса погоды.
        {
            string weather_descript;

            Net.Weather();

            MaterialLabelWindSpeed.Text = Net.GetWeather("wind") + "м/с";
            MaterialLabelDegWind.Text = Net.GetWeather("wind_deg") + "°";

            weather = Net.GetWeather("weather");

            weather_descript = Net.GetWeather("weather_long");
            MaterialLabelWeather.Text = weather_descript.ToUpper();
            switch (weather)
            {
                case "Clouds":
                    PictureBoxWeather.Image = Properties.Resources.облако_96;
                    break;
                case "Partly Cloudy":
```

```
                    PictureBoxWeather.Image =
Properties.Resources.частичная_облачность_96;
                    break;
                case "Fog":
                    PictureBoxWeather.Image =
Properties.Resources.туман_днем_96;
                    break;
                case "Extreme":
                    PictureBoxWeather.Image =
Properties.Resources.шторм_с_проливным_дождем_96;
                    break;
                case "Cleanly":
                    PictureBoxWeather.Image = Properties.Resources.солнечно_96;
                    break;
                case "Rain":
                    PictureBoxWeather.Image = Properties.Resources.ливень_96;
                    break;
                case "Wet Snow":
                    PictureBoxWeather.Image =
Properties.Resources.мокрый_снег_96;
                    break;
                case "Snow":
                    PictureBoxWeather.Image = Properties.Resources.снег_96;
                    break;
            }
            TimerGetWither.Interval = 1200000;
        }

        void ap_db()
        {
            //if (ADC_conect == true)
            //{
            sqlConnection.Open();
            if (sqlConnection.State == ConnectionState.Open)
            {
                //для теста
                Random rnd = new Random();
                int volteg = rnd.Next(20, 31);
                int curent = rnd.Next(10, 50);
                int power = volteg * curent;
                int angel = 60;

                SqlCommand command = new SqlCommand($"INSERT INTO [DataGraf]
(time, volteg_solar, power, Angle, curent) VALUES
(N'{DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss")}', N'{volteg}', N'{power}',
N'{angel}', N'{curent}')", sqlConnection); //volteg, curent, power
                command.ExecuteNonQuery();
                sqlConnection.Close();
            }
            //}
        }


        private void timer_write_text_file_Tick(object sender, EventArgs e)
        {
            File.ParsingWriteText(power, volteg, curent, angle);
        }

        private void timer_blink_Tick(object sender, EventArgs e) // Таймер 2
отвечает за мегание светодиодами.
        {
            if (C == 0)
            {
                if (led_blink_set == 1)
```

```
            {
                HardwareMCP2210.ControlPorts(1, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.включить_свет;
            }
            else if (led_blink_set == 2)
            {
                HardwareMCP2210.ControlPorts(1, 101, modul_box);
                pictureLamp2.Image = Properties.Resources.включить_свет;
            }
            else if (led_blink_set == 3)
            {
                HardwareMCP2210.ControlPorts(1, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.включить_свет;
                HardwareMCP2210.ControlPorts(1, 101, modul_box);
                pictureLamp2.Image = Properties.Resources.включить_свет;
            }
            C = 1;
        }
        else if (C == 1)
        {
            if (led_blink_set == 1)
            {
                HardwareMCP2210.ControlPorts(0, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.выключить_свет;
            }
            else if (led_blink_set == 2)
            {
                HardwareMCP2210.ControlPorts(0, 101, modul_box);
                pictureLamp2.Image = Properties.Resources.выключить_свет;
            }
            else if (led_blink_set == 3)
            {
                HardwareMCP2210.ControlPorts(0, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.выключить_свет;
                HardwareMCP2210.ControlPorts(0, 101, modul_box);
                pictureLamp2.Image = Properties.Resources.выключить_свет;
            }
            C = 0;
        }
    }


    private void Control_Server_Setting()
    {
        if (control_server)
        {
            if (response_server == "start1")
            {
                if (led_blink_set != 0 && led_blink_set != 2)
                {
                    led_blink_set -= 1;
                }
                HardwareMCP2210.ControlPorts(1, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.включить_свет;
            }
            else if (response_server == "stop1")
            {
                if (led_blink_set != 0 && led_blink_set != 2)
                {
                    led_blink_set -= 1;
                }
                HardwareMCP2210.ControlPorts(0, 100, modul_box);
                pictureLamp1.Image = Properties.Resources.выключить_свет;
```

```
                }
                else if (response_server == "blink1")
                {
                    if (led_blink_set == 0 || led_blink_set == 2)
                    {
                        led_blink_set += 1;
                    }
                    timer_blink.Enabled = true;
                }
                ////////////////////////////////////////////////////////////
Второй светодиод.

                if (response_server == "start2")
                {
                    if (led_blink_set > 1)
                    {
                        led_blink_set -= 2;
                    }
                    HardwareMCP2210.ControlPorts(1, 101, modul_box);
                    pictureLamp2.Image = Properties.Resources.включить_свет;
                }
                else if (response_server == "stop2")
                {
                    if (led_blink_set > 1)
                    {
                        led_blink_set -= 2;
                    }
                    HardwareMCP2210.ControlPorts(0, 101, modul_box);
                    pictureLamp2.Image = Properties.Resources.выключить_свет;
                }
                else if (response_server == "blink2")
                {
                    if (led_blink_set == 0 || led_blink_set == 1)
                    {
                        led_blink_set += 2;
                    }
                    timer_blink.Enabled = true;
                }
            }
            else if (control_server == false)
            {
                timer_blink.Enabled = false;
                HardwareMCP2210.ControlPorts(0, 0, modul_box);
                ColorReset();
            }
            else
            {

            }
        }

        ////////////////////////////////////////Панель
Елементов//////////////////////////////////////////
        private void истроияВерсийToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (FormInfo == null || FormInfo.IsDisposed)
            {
                FormInfo = new FormInfoProgram();
                FormInfo.Show();
            }
            else
            {
                FormInfo.Activate();
```

```
            }
        }
        private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            SoundPlayer info = new SoundPlayer(Properties.Resources.info);
            MaterialSnackBar SnackBarMessage = new MaterialSnackBar("Версія -
X.X.X | Реліз 010622", "OK", true);
            info.Play();
            SnackBarMessage.Show(this);
        }
        private void настройкиToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            if (SettForm == null || SettForm.IsDisposed)
            {
                SettForm = new FormSetting();
                SettForm.Show();
                //SettForm.FormClosed += new
FormClosedEventHandler(NotifyAboutClosedSettForm);
            }
            else
            {
                SettForm.Activate();
            }
        }
        private void окноСГрафикоммиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (GrafShow == null || GrafShow.IsDisposed)
            {
                GrafShow = new GrafForm();
                GrafShow.Show();
            }
            else
            {
                GrafShow.Activate();
            }
        }
        private void обновлениеToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            if (FormUpdt == null || FormUpdt.IsDisposed)
            {
                FormUpdt = new FormUpdt();
                FormUpdt.FormClosed += new
FormClosedEventHandler(NotifyAboutClosedChildForm);
                FormUpdt.Show();
            }
            else
            {
                FormUpdt.Activate();
            }
        }

        ////////////////////////////////////////Событие
формы//////////////////////////////////////////
        private void Main_activ(object sender, EventArgs e)
        {
            activ_report_solar =
Convert.ToBoolean(Settings.Default["activ_report_solar"]);
            GET_interval = Convert.ToInt32(Settings.Default["polling_time"]);
            time_out = Convert.ToInt32(Settings.Default["time_out"]);
            dark_theme = Convert.ToBoolean(Settings.Default["dark_theme"]);
```

```
            activ_log = Convert.ToBoolean(Settings.Default["activ_log"]);
            //MQTT
            portMQTT =
Convert.ToInt32(Settings.Default["mqtt_port"].ToString());
            adress_brokerMQTT = Settings.Default["mqtt_ip"].ToString();
            loginMQTT = Settings.Default["mqtt_login"].ToString();
            passwordMQTT = Settings.Default["mqtt_password"].ToString();
            //
            if (dark_theme)
            {
                this.BackColor = System.Drawing.Color.Gray;
            }
            else
            {
                this.BackColor = System.Drawing.Color.WhiteSmoke;
            }
        }
        private void FormMain_Load(object sender, EventArgs e)
        {
            Size resolution = Screen.PrimaryScreen.Bounds.Size;
            this.Location = new Point(0, 0);
            this.StartPosition = FormStartPosition.Manual;
            //this.Size = resolution;

            NotificationArea.BalloonTipText = "Приложение свёрнуто.";

            // Подключение к базе данных.
            DatabaseLink = "Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=" +
Directory.GetCurrentDirectory() + "\\Database.mdf;Integrated Security=True";
            //ConfigurationManager.ConnectionStrings["DB"].ConnectionString
            try
            {
                sqlConnection = new SqlConnection(DatabaseLink);
                sqlConnection.Open();
                if (sqlConnection.State == ConnectionState.Open)
                {
                    File.FileWriteDataLog("Database connected", "info",
activ_log);
                    sqlConnection.Close();
                }
            }
            catch
            {
                File.FileWriteDataLog("Database not connected", "error",
activ_log);
            }
        }
        private void NotifyAboutClosedChildForm(object sender,
FormClosedEventArgs args)
        {
            if (File.ChekNewProgramStart())
            {
                this.Close();
            }
        }
        private void FormMain_Resize(object sender, EventArgs e)
        {
            if (WindowState == FormWindowState.Minimized)
            {
                this.Hide();
                NotificationArea.ShowBalloonTip(1000);
            }
        }
```

```
        private void FormMain_FormClosed(object sender, FormClosedEventArgs e)
        {
            MqttThread.Abort();
            MqttThread.Join(1000);
            GetThread.Abort();
            GetThread.Join(1000);
            if (use_local_broker)
            {
                netMQTT.ExitBroker();
            }
            netMQTT.DisconentMQTT();
        }
        private void PictureBoxSolarPanel_Paint(object sender, PaintEventArgs e)
        {
            e.Graphics.TranslateTransform(220, 280);
            e.Graphics.RotateTransform(((float)construction_angle - 90));
            e.Graphics.ScaleTransform((float)2, (float)2);
            e.Graphics.DrawImage(image_solat_panel, -75, -15);
        }


////////////////////////////////////Трей/////////////////////////////////////
/////
        private void NotificationArea_MouseDoubleClick(object sender,
MouseEventArgs e)
        {
            this.Show();
            WindowState = FormWindowState.Normal;
        }
        private void закрытьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
    public class HardwareMCP2210
    {
        /*
        UsbDisconect - Отключает модуль с указанным ID. Возвращает: true -
отключено, false - модуля не существует или уже был отключён.
        UsbCheckConnection - Проверяет связь с уже подключенными модулями,если
auto_disconnect = true удаляет модуль с которым утрачено подключение.
Возвращает: строку ошибки.
        UsbConnect - Подключает модуль с указанным PID. Возвращает: строку
состояния, и ID модуля.
        ControlPorts - Управление портами, принимает статус порта 1 - вкл, 0 -
выкл. Код порта 100 - GP0, 101 - GP1, 102 - GP2... Спец код 0 - отключить все
порты. ID модуля.
        ADCDataReception - Опрашивает АЦП, принимает ID модуля к которому
подключен АЦП и массив с данными для возврата считаных данных.
        StepMotorControl -  Управляет шаговим мотором. Принимает: set_angle -
угол который необходимо достичь, cur_angle - усреднённый угол.
        */
        bool[] port_arr_controll = { false, false, false, false, false, false,
false, false }; //Тригеры срабатывания портов.
        byte on_off_bit_port = 0; //Бит порта. Используется для управления
выводами.
        byte index_arr_pid = 0;
        const ushort base_vid = 0x4D8;
        ushort[] pid_modul = new ushort[5];

        public bool UsbDisconect(IntPtr mcp_module)
        {
            ushort VID = 0, PID = 0, power_limit = 0;
            byte power_source = 0, power_wake_up = 0;
```

```
            try
            {
                MCP2210.M_Mcp2210_GetUsbKeyParams(mcp_module, ref VID, ref PID,
ref power_wake_up, ref power_source, ref power_limit);
                MCP2210.M_Mcp2210_Close(mcp_module);
                DelPid(PID);
                return true;
            }
            catch
            {
                return false;
            }
        }
        public string UsbCheckConnection(bool auto_disconnect)
        {
            byte while_counter = 0;
            ushort[] lost_pid = new ushort[5];
            int connect_error;

            while (while_counter < 5)
            {
                if (pid_modul[while_counter] != 0)
                {
                    connect_error =
MCP2210.M_Mcp2210_GetConnectedDevCount(base_vid, pid_modul[while_counter]);
                    if (connect_error == 0)
                    {
                        lost_pid[while_counter] = pid_modul[while_counter];
                        if (auto_disconnect)
                        {
                            DelPid(pid_modul[while_counter]);
                        }
                        return ("Lost connection modul VID-" +
(base_vid.ToString()) + " PID-" + (pid_modul[while_counter].ToString()));
                    }
                }
                lost_pid[while_counter] = 0;
                while_counter++;
            }
            return "All conect";
        }
        public string UsbConnect(ushort PID, ref IntPtr mcp_module)
        {
            //стандартные VID и PID у модуля MCP2210 0x4D8 | 0xDE
            int connect_error_ST =
MCP2210.M_Mcp2210_GetConnectedDevCount(base_vid, PID);
            if (connect_error_ST == 0)
            {
                return "Module not found";
            }
            else if (pid_modul.Contains(PID))
            {
                return "Module already connected";
            }
            else
            {
                StringBuilder Path = null;
                mcp_module = MCP2210.M_Mcp2210_OpenByIndex(base_vid, PID, 0,
Path);

                //скорость передачи данных
                uint rate = 9600;
                //настройка выводов CS
                uint cs_idle = 0xFFF;
```

```
                uint cs_activ = 0xFBF;
                //задержки в микросекундах.
                uint cs_to_data_dly = 0;
                uint data_to_data_dly = 0;
                uint data_to_cs_dly = 0;
                //количество байт передачи/приема
                uint numberbyte = 3;
                byte spi_mode = (byte)MCP2210.M_MCP2210_SPI_MODE0;
                byte[] Tx = new byte[3];
                byte[] Rx = new byte[3];
                byte[] gpio = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00,
0x00 }; // 0-gpio 1-spi  2 -indication
                uint run_state_pin = 0; //511 -run prg with state 1
                uint dir_pin_state = 0; // 0-out 1-input

                //настраиваем GPIO
                MCP2210.M_Mcp2210_SetGpioConfig(mcp_module,
(byte)MCP2210.M_MCP2210_VM_CONFIG, gpio, run_state_pin, dir_pin_state,
                    (byte)MCP2210.M_MCP2210_REMOTE_WAKEUP_DISABLED,
(byte)MCP2210.M_MCP2210_INT_MD_CNT_NONE,
                    (byte)MCP2210.M_MCP2210_SPI_BUS_RELEASE_DISABLED);
                MCP2210.M_Mcp2210_SetSpiConfig(mcp_module,
(byte)MCP2210.M_MCP2210_VM_CONFIG, ref rate, ref cs_idle, ref cs_activ, ref
cs_to_data_dly, ref data_to_data_dly, ref data_to_cs_dly, ref numberbyte, ref
spi_mode);

                AddPid(PID);
                return "Module connect PID " + (PID.ToString());
            }
        }
        private void AddPid(ushort PID)
        {
            if (!pid_modul.Contains(PID))
            {
                pid_modul[index_arr_pid] = PID;
                index_arr_pid++;
                if (index_arr_pid > 4)
                {
                    index_arr_pid = 0;
                }
            }
        }
        private void DelPid(ushort PID)
        {
            if (pid_modul.Contains(PID))
            {
                pid_modul[Array.IndexOf(pid_modul, PID)] = 0;
            }
        }
        public void ControlPorts(byte status_port, byte cod_port, IntPtr
mcp_module)
        {
            uint mask = 0;
            if (cod_port == 100)
            {
                if (status_port == 1 && port_arr_controll[0] == false)
                {
                    on_off_bit_port += 1;
                    port_arr_controll[0] = true;
                }
                else if (status_port == 0 && port_arr_controll[0] == true)
                {
                    on_off_bit_port -= 1;
                    port_arr_controll[0] = false;
```

```
    }
}
else if (cod_port == 101)
{
    if (status_port == 1 && port_arr_controll[1] == false)
    {
        on_off_bit_port += 2;
        port_arr_controll[1] = true;
    }
    else if (status_port == 0 && port_arr_controll[1] == true)
    {
        on_off_bit_port -= 2;
        port_arr_controll[1] = false;
    }
}
else if (cod_port == 102)
{
    if (status_port == 1 && port_arr_controll[2] == false)
    {
        on_off_bit_port += 4;
        port_arr_controll[2] = true;
    }
    else if (status_port == 0 && port_arr_controll[2] == true)
    {
        on_off_bit_port -= 4;
        port_arr_controll[2] = false;
    }
}
else if (cod_port == 103)
{
    if (status_port == 1 && port_arr_controll[3] == false)
    {
        on_off_bit_port += 8;
        port_arr_controll[3] = true;
    }
    else if (status_port == 0 && port_arr_controll[3] == true)
    {
        on_off_bit_port -= 8;
        port_arr_controll[3] = false;
    }
}
else if (cod_port == 104)
{
    if (status_port == 1 && port_arr_controll[3] == false)
    {
        on_off_bit_port += 16;
        port_arr_controll[4] = true;
    }
    else if (status_port == 0 && port_arr_controll[3] == true)
    {
        on_off_bit_port -= 16;
        port_arr_controll[4] = false;
    }
}
else if (cod_port == 105)
{
    if (status_port == 1 && port_arr_controll[3] == false)
    {
        on_off_bit_port += 32;
        port_arr_controll[5] = true;
    }
    else if (status_port == 0 && port_arr_controll[3] == true)
    {
        on_off_bit_port -= 32;
```

```
                    port_arr_controll[5] = false;
                }
            }
            else if (cod_port == 106)
            {
                if (status_port == 1 && port_arr_controll[3] == false)
                {
                    on_off_bit_port += 64;
                    port_arr_controll[6] = true;
                }
                else if (status_port == 0 && port_arr_controll[3] == true)
                {
                    on_off_bit_port -= 64;
                    port_arr_controll[6] = false;
                }
            }
            else if (cod_port == 107)
            {
                if (status_port == 1 && port_arr_controll[7] == false)
                {
                    on_off_bit_port += 128;
                    port_arr_controll[7] = true;
                }
                else if (status_port == 0 && port_arr_controll[7] == true)
                {
                    on_off_bit_port -= 128;
                    port_arr_controll[7] = false;
                }
            }
            else if (cod_port == 0)
            {
                on_off_bit_port = 0;
                port_arr_controll[0] = false;
                port_arr_controll[1] = false;
                port_arr_controll[2] = false;
                port_arr_controll[3] = false;
                port_arr_controll[4] = false;
                port_arr_controll[5] = false;
                port_arr_controll[6] = false;
                port_arr_controll[7] = false;
            }
            try
            {
                MCP2210.M_Mcp2210_SetGpioPinVal(mcp_module, on_off_bit_port, ref
mask);
            }
            catch
            {

            }
        }
        public void ADCDataReception(IntPtr mcp_module, ref double[] data_value)
        {
            int error = 0;
            uint rate = 9600;
            uint numberbyte = 3;
            Double current_factor =
Convert.ToDouble(Settings.Default["current_measurement_factor"]);
            Double voltage_factor =
Convert.ToDouble(Settings.Default["voltage_measurement_factor"]);
            Double angle_factor =
Convert.ToDouble(Settings.Default["angle_measurement_factor"]);

            double volteg, curent, angle;
```

```
            byte[] Tx = new byte[3];
            byte[] Rx = new byte[3];
            short temp = 0;
            Tx[0] = 0b00001100;
            Tx[1] = 0b00000000;
            Tx[2] = 0b00000000;

            error = MCP2210.M_Mcp2210_xferSpiData(mcp_module, Tx, Rx, ref rate,
ref numberbyte, 0x80000000);
            if (error != 0)
            {
                /*
                curent = double.NaN;
                volteg = double.NaN;
                angle = double.NaN;
                */
                curent = 35;
                volteg = 33.4;
                angle = 60;
                data_value[0] = volteg;
                data_value[1] = curent;
                data_value[2] = angle;
            }
            else
            {
                Rx[0] = Rx[2];
                Rx[1] = Convert.ToByte(Rx[1] & 0b00001111);
                temp = BitConverter.ToInt16(Rx, 0);

                angle = temp * 0.0006103515625 * angle_factor;

////////////////////////////////////////////////////////////////////////////////
////////////////////////
                Tx[1] = 0b10000000;

                MCP2210.M_Mcp2210_xferSpiData(mcp_module, Tx, Rx, ref rate, ref
numberbyte, 0x80000000);
                Rx[0] = Rx[2];
                Rx[1] = Convert.ToByte(Rx[1] & 0b00001111);
                temp = BitConverter.ToInt16(Rx, 0);

                volteg = temp * 0.0006103515625 * voltage_factor;

////////////////////////////////////////////////////////////////////////////////
////////////////////////

                Tx[0] = 0b00001101;
                Tx[1] = 0b00000000;

                MCP2210.M_Mcp2210_xferSpiData(mcp_module, Tx, Rx, ref rate, ref
numberbyte, 0x80000000);
                Rx[0] = Rx[2];
                Rx[1] = Convert.ToByte(Rx[1] & 0b00001111);
                temp = BitConverter.ToInt16(Rx, 0);

                curent = (((temp * 0.0006103515625) / 0.04) - 50) *
current_factor;

////////////////////////////////////////////////////////////////////////////////
////////////////////////
                data_value[0] = volteg;
                data_value[1] = curent;
                data_value[2] = angle;
            }
```

```
        }
        public string StepMotorControl(double set_angle, double cur_angle,
IntPtr mcp_module)
        {
            double[] buffer_adc = new double[3];
            int steps, steps_text;
            double angle_dif = set_angle - cur_angle;
            if (angle_dif > 0)
            {
                ControlPorts(1, 104, mcp_module);

                steps = Convert.ToInt32(Math.Abs(angle_dif / 0.18));
                steps_text = steps;

                while (steps < 0)
                {
                    ControlPorts(1, 105, mcp_module);
                    ControlPorts(0, 105, mcp_module);
                    ADCDataReception(mcp_module, ref buffer_adc);
                    if (Math.Round(buffer_adc[2], 2) == cur_angle ||
buffer_adc[2] == double.NaN)
                    {
                        return "Motion error!";
                    }
                    steps--;
                }
                return "Movement successful. Number of steps :" +
steps_text.ToString() + ".";
            }
            else if (angle_dif < 0)
            {
                ControlPorts(0, 104, mcp_module);

                steps = Convert.ToInt32(Math.Abs(angle_dif / 0.18));
                steps_text = steps;

                while (steps < 0)
                {
                    ControlPorts(1, 105, mcp_module);
                    ControlPorts(0, 105, mcp_module);
                    ADCDataReception(mcp_module, ref buffer_adc);
                    if (Math.Round(buffer_adc[2], 2) == cur_angle ||
buffer_adc[2] == double.NaN)
                    {
                        return "Motion error!";
                    }
                    steps--;
                }
                return "Movement successful. Number of steps :" +
steps_text.ToString() + ".";
            }
            return "Unknown parameter movement not possible.";
        }
    }

    public class Net
    {
        Rootobject responsWeather;

        public bool Transmitter(ref string respons_server, string url_server) //
обработчик запросов
        {
            HttpWebRequest myHttpWebRequest =
(HttpWebRequest)HttpWebRequest.Create(url_server);
```

```
            myHttpWebRequest.Headers["Accept-Language"] = "ru-RU,ru;q=0.8,en-
US;q=0.6,en;q=0.4";
            myHttpWebRequest.Headers["Accept-Charset"] = "iso-8859-1,*,utf-8";
            myHttpWebRequest.Headers["Upgrade-Insecure-Requests"] = "1";
            myHttpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 6.1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36";
            try
            {
                WebResponse myHttpWebResponse =
myHttpWebRequest.GetResponse();//Отправляем данные и получаем ответ
                Stream dataStream =
myHttpWebResponse.GetResponseStream();//Получаем поток ответа
                StreamReader reader = new StreamReader(dataStream);//Новый
объект для чтения потока
                respons_server = reader.ReadToEnd();//Считываем в строку весь
ответ сервера

                reader.Close();//закрываем "чтеца"
                dataStream.Close();//закрываем поток ответа
                myHttpWebResponse.Close();//И запрос.
                return true;
            }
            catch
            {
                return false;
            }
        }

        public string GetWeather(string parametr)
        {
            switch (parametr)
            {
                case "wind":
                    return responsWeather.wind.speed.ToString();
                case "weather":
                    return responsWeather.weather[0].main.ToString();
                case "weather_long":
                    return responsWeather.weather[0].description.ToString();
                case "wind_deg":
                    return responsWeather.wind.deg.ToString();
            }
            return "ERROR";
        }
        public void Weather()
        {
            try
            {
                string url_weather_sourse =
"https://api.openweathermap.org/data/2.5/weather?lat=48.45&lon=35&appid=275731b3
3265047fd805e30b9ffac55e&units=metric&lang=ru";
                HttpWebRequest httpWebRequest =
(HttpWebRequest)WebRequest.Create(url_weather_sourse);
                HttpWebResponse httpWebResponse =
(HttpWebResponse)httpWebRequest.GetResponse();

                using (StreamReader streamReader = new
StreamReader(httpWebResponse.GetResponseStream()))
                {
                    responsWeather =
JsonConvert.DeserializeObject<Rootobject>(streamReader.ReadToEnd());
                }
            }
            catch
            {
```

```
            }
        }
    }
    public class NetMQTT
    {
        Process iStartProcess = new Process();
        MqttClientOptionsBuilder builder;
        ManagedMqttClientOptions options;
        MqttApplicationMessage applicationMessage;
        IManagedMqttClient _mqttClient;
        public void InitializationMQTT(string host, int port)
        {
            builder = new MqttClientOptionsBuilder()
                                        .WithClientId(Guid.NewGuid().ToString())
                                        .WithTcpServer(host, port);

            options = new ManagedMqttClientOptionsBuilder()
                        .WithAutoReconnectDelay(TimeSpan.FromSeconds(60))
                        .WithClientOptions(builder.Build())
                        .Build();

            _mqttClient = new MqttFactory().CreateManagedMqttClient();
        }
        public bool PublishData(double[] parameters, string[] data_android)
        {
            if (!_mqttClient.IsStarted)
            {
                _mqttClient.StartAsync(options).GetAwaiter().GetResult();
            }
            if (_mqttClient.IsConnected)
            {
                //"volteg solar", "curent load", "power load", "volteg battery",
"curent load 2", "power load 2", "power load all"
                string json = JsonConvert.SerializeObject(new
                {
                    volteg_solar = parameters[0],
                    volteg_bat = parameters[3],
                    curent1 = parameters[1],
                    curent2 = parameters[4],
                    power1 = parameters[2],
                    power2 = parameters[5],
                    power_all = parameters[6],
                    data = DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss")
                });
                _mqttClient.PublishAsync("PC/parameters", json);

                json = JsonConvert.SerializeObject(new
                {
                    Load1 = data_android[0],
                    Load2 = data_android[1],
                    RemoteControlLoad = data_android[2],
                    AutomaticControl = data_android[3],
                    TiltAngle = data_android[4],
                    data = DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss")
                });
                _mqttClient.PublishAsync("PC/sys/status", json);

                return true;
            }
            else
            {
                return false;
            }
```

```
        }
        public void DisconentMQTT()
        {
            if (_mqttClient.IsStarted)
            {
                _mqttClient.UnsubscribeAsync();
                _mqttClient.StopAsync();
            }
        }


        public void StartBroker()
        {
            iStartProcess.StartInfo.FileName = @"C:\Program Files
(x86)\mosquitto\mosquitto.exe";
            iStartProcess.StartInfo.Arguments = " -v";
            //iStartProcess.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            iStartProcess.Start();
        }
        public void ExitBroker()
        {
            iStartProcess.Close();
        }
    }
    public class SQLDataBase
    {

    }
    public class FileManager : Form
    {
        string temp_name_data_solar;
        public void FileMain()
        {
            if (CatalogChek())
            {
                ChekOrCreat();
            }
        }
        public void FileWriteDataLog(string data_log, string typ, bool
activ_log) // примитивный логер.
        {
            string temp;
            temp = typ + " " + DateTime.Now.ToLongDateString() + " | " +
DateTime.Now.ToLongTimeString() + " ->" + data_log + "<-\n";
            if (typ == "error" && activ_log)
            {
                File.AppendAllText("Data SPCS\\data_log.txt", temp);
            }
            else if (typ == "info" && activ_log)
            {
                File.AppendAllText("Data SPCS\\data_log.txt", temp);
            }
            else if (typ == "fatal" && activ_log)
            {
                File.AppendAllText("Data SPCS\\data_log.txt", temp);
            }
            ChekSizeLog();
        }
        bool CatalogChek() // Проверяет наличие каталогов, если отсутствуют
создаёт.
        {
            bool Catalog_creat;
            Catalog_creat = (Directory.Exists("Data SPCS") &&
Directory.Exists("Update") && Directory.Exists("SPCS Solar data"));
```

```
            if (Catalog_creat == false)
            {
                Directory.CreateDirectory("Data SPCS");
                Directory.CreateDirectory("SPCS Solar Data");
                Directory.CreateDirectory("Update");
                Catalog_creat = true;
            }
            return Catalog_creat;
        }
        public void ParsingWriteText(double power, double volteg, double curent,
double angle) //Записывает файл с данными согласно выбранному порядку.
        {
            string data_temp =
Convert.ToString(Settings.Default["type_path_in_file"]);
            if (data_temp.Contains("{TIME}"))
            {
                data_temp = data_temp.Replace("{TIME}",
DateTime.Now.ToShortTimeString().Replace(':', '.'));
            }
            if (data_temp.Contains("{POWER}"))
            {
                data_temp = data_temp.Replace("{POWER}",
Convert.ToString(power).Replace(',', '.'));
            }
            if (data_temp.Contains("{VOLTAGE}"))
            {
                data_temp = data_temp.Replace("{VOLTAGE}",
Convert.ToString(volteg).Replace(',', '.'));
            }
            if (data_temp.Contains("{CURRENT}"))
            {
                data_temp = data_temp.Replace("{CURRENT}",
Convert.ToString(curent).Replace(',', '.'));
            }
            if (data_temp.Contains("{ANGLE}"))
            {
                data_temp = data_temp.Replace("{ANGLE}",
Convert.ToString(angle).Replace(',', '.'));
            }
            if (data_temp.Contains("{WORK_MOD}"))
            {
                data_temp = data_temp.Replace("{WORK_MOD}",
Convert.ToString(angle).Replace(',', '.'));
            }
            data_temp = data_temp + "\n";
            string new_temp_name_data_solar = "SPCS Solar Data" + "\\" +
("Solar_info_on_day" + DateTime.Now.ToString("(yyyy_MM_dd)") + ".txt");
            if (temp_name_data_solar != new_temp_name_data_solar)
            {
                temp_name_data_solar = new_temp_name_data_solar;
                File.AppendAllText(temp_name_data_solar,
(Convert.ToString(Settings.Default["type_path_in_file"]) + "\n"));
            }
            File.AppendAllText(temp_name_data_solar, data_temp);
        }
        void ChekOrCreat()
        {
            try
            {
                FileStream datalog = new FileStream("Data SPCS\\data_log.txt",
FileMode.Open);
                datalog.Close();
            }
            catch
```

```
            {
                File.AppendAllText("Data SPCS\\data_log.txt", "");
            }
            try
            {
                FileStream dataup = new FileStream("Data SPCS\\data_up.txt",
FileMode.Open);
                dataup.Close();
            }
            catch
            {
                File.AppendAllText("Data SPCS\\data_up.txt", "");
            }
            try
            {
                temp_name_data_solar = "SPCS Solar Data" + "\\" +
("Solar_info_on_day" + DateTime.Now.ToString("(yyyy_MM_dd)") + ".txt");
                FileStream solar_data = new FileStream(temp_name_data_solar,
FileMode.Open);
                solar_data.Close();
            }
            catch
            {
                temp_name_data_solar = "SPCS Solar Data" + "\\" +
("Solar_info_on_day" + DateTime.Now.ToString("(yyyy_MM_dd)") + ".txt");
                File.AppendAllText(temp_name_data_solar,
(Convert.ToString(Settings.Default["type_path_in_file"]) + "\n"));
            }
        }
        public bool ChekNewProgramStart()
        {
            if (File.ReadAllText("Data SPCS\\data_up.txt") == "1")
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        void ChekSizeLog()
        {
            int max_size = Convert.ToInt32(Settings.Default["log_length"]);
            int size_log = File.ReadAllLines("Data SPCS\\data_log.txt").Length;
            String[] temp;
            if (max_size <= size_log)
            {
                temp = File.ReadAllLines("Data SPCS\\data_log.txt");
                var n_temp = temp.Skip(100);
                File.WriteAllLines("Data SPCS\\data_log.txt", n_temp);
                FileWriteDataLog("Removed the first 20 entries", "info", true);
            }
        }
        public void ClearUpdate()
        {
            if (ChekNewProgramStart())
            {
                File.Delete(@"Update\SPCS.exe");
                File.Delete(@".\UPDATE.zip");
                File.WriteAllText("Data SPCS\\data_up.txt", "");
            }
        }
    }
```

```
//////////////////////////////////////////////////////////////////////////////
///
    public class Rootobject
    {
        public Coord coord { get; set; }
        public Weather[] weather { get; set; }
        public string _base { get; set; }
        public Main main { get; set; }
        public int visibility { get; set; }
        public Wind wind { get; set; }
        public Clouds clouds { get; set; }
        public int dt { get; set; }
        public Sys sys { get; set; }
        public int timezone { get; set; }
        public int id { get; set; }
        public string name { get; set; }
        public int cod { get; set; }
    }
    public class Coord
    {
        public float lon { get; set; }
        public float lat { get; set; }
    }
    public class Main
    {
        public float temp { get; set; }
        public float feels_like { get; set; }
        public float temp_min { get; set; }
        public float temp_max { get; set; }
        public int pressure { get; set; }
        public int humidity { get; set; }
    }
    public class Wind
    {
        public float speed { get; set; }
        public int deg { get; set; }
    }
    public class Clouds
    {
        public int all { get; set; }
    }
    public class Sys
    {
        public int type { get; set; }
        public int id { get; set; }
        public float message { get; set; }
        public string country { get; set; }
        public int sunrise { get; set; }
        public int sunset { get; set; }
    }
    public class Weather
    {
        public int id { get; set; }
        public string main { get; set; }
        public string description { get; set; }
        public string icon { get; set; }
    }

}
```

Форма FormGraf

```
////////////////////////////////////////////////////////////////////////////////
public partial class GrafForm : MaterialForm
    {
        ChartValues<int> volteg_solar = new ChartValues<int>();
        ChartValues<int> power = new ChartValues<int>();
        ChartValues<int> angle = new ChartValues<int>();
        ChartValues<int> curent = new ChartValues<int>();
        List<string> dates = new List<string>();

        SeriesCollection series = new SeriesCollection();
        SqlConnection sqlConnection;

        string program_directory = Directory.GetCurrentDirectory();
        int buffer_real_time;
        byte zoom;
        bool access = false;

        public GrafForm()
        {
            InitializeComponent();

            var mapper = Mappers.Xy<MeasureModel>()
                .X(model => model.DateTime.Ticks)
                .Y(model => model.Value);
            Timer = new Timer
            {
                Interval = 1000
            };
            Timer.Tick += TimerOnTick;

            Charting.For<MeasureModel>(mapper);
            ChartValues = new ChartValues<MeasureModel>();
            ChartValues1 = new ChartValues<MeasureModel>();
            ChartValues2 = new ChartValues<MeasureModel>();
            ChartValues3 = new ChartValues<MeasureModel>();
            R = new Random();

            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.EnforceBackcolorOnAllComponents = true;
            materialSkinManager.AddFormToManage(this);
            GrafUpdate(true, false);
        }

        private void GrafForm_Load(object sender, EventArgs e)
        {
            if (Convert.ToBoolean(Settings.Default["dark_theme"]))
            {
                cartesianChart.Background = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(46, 46, 46));
            }
            else
            {
                cartesianChart.Background = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(245, 245, 245));
            }
            SqlDataBaseLoad();
        }

        void SqlDataBaseLoad()
        {
```

```
            sqlConnection = new SqlConnection("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=" +
Directory.GetCurrentDirectory() + "\\Database.mdf;Integrated Security=True");
            sqlConnection.Open();
            if (sqlConnection.State == ConnectionState.Open)
            {
                string sql_expression = "SELECT * FROM DataGraf";
                SqlCommand command = new SqlCommand(sql_expression,
sqlConnection);
                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        dates.Add(Convert.ToString(reader.GetValue(1)));
                        volteg_solar.Add(Convert.ToInt32(reader.GetValue(2)));
                        power.Add(Convert.ToInt32(reader.GetValue(3)));
                        angle.Add(Convert.ToInt32(reader.GetValue(4)));
                        curent.Add(Convert.ToInt32(reader.GetValue(5)));
                    }
                }
                sqlConnection.Close();
            }
            else
            {
            }
        }
        void GrafUpdate(bool first_start, bool real_time_mode)
        {
            var gradientBrush = new LinearGradientBrush
            {
                StartPoint = new System.Windows.Point(0, 0),
                EndPoint = new Point(0, 1)
            };
            gradientBrush.GradientStops.Add(new
GradientStop(System.Windows.Media.Color.FromRgb(60, 179, 113), 0));
            gradientBrush.GradientStops.Add(new GradientStop(Colors.Transparent,
1));

            if (first_start)
            {
                cartesianChart.AxisX.Clear();
                cartesianChart.AxisY.Clear();
                cartesianChart.Series.Clear();
                SqlDataBaseLoad();
            }
            if (!real_time_mode)
            {
                Timer.Stop();
                Timer.Enabled = false;
                while (ChartValues.Count > 0) ChartValues.RemoveAt(0);

                if (materialCheckbox_vs_graf_visible.Checked)
                {
                    cartesianChart.Series.Add(new LineSeries
                    {
                        Values =
volteg_solar.AsGearedValues().WithQuality(Quality.Low),
                        Name = "Напруга",
                        Title = "Напруга СП",
                        StrokeThickness = 5,
                        StrokeDashArray = new
System.Windows.Media.DoubleCollection(50),
```

```
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(60, 179, 113)),
                        Fill = gradientBrush,
                        PointGeometrySize = 5
                    });
                }
                if (materialCheckbox_p1_graf_visible.Checked)
                {
                    cartesianChart.Series.Add(new LineSeries
                    {
                        Values =
power.AsGearedValues().WithQuality(Quality.Low),
                        Name = "Потужність",
                        Title = "Потужність",
                        StrokeThickness = 5,
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(123, 104, 238)),
                        Fill = Brushes.Transparent,
                        PointGeometrySize = 5,
                        PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                    });
                }
                if (materialCheckbox_p2_graf_visible.Checked)
                {
                    cartesianChart.Series.Add(new LineSeries
                    {
                        Values =
angle.AsGearedValues().WithQuality(Quality.Low),
                        Name = "Кут",
                        Title = "Кут",
                        StrokeThickness = 5,
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(72, 61, 139)),
                        Fill = Brushes.Transparent,
                        PointGeometrySize = 5,
                        PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                    });
                }
                if (materialCheckbox_c1_graf_visible.Checked)
                {
                    cartesianChart.Series.Add(new LineSeries
                    {
                        Values =
curent.AsGearedValues().WithQuality(Quality.Low),
                        Name = "Струм",
                        Title = "Струм",
                        StrokeThickness = 5,
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(255, 99, 71)),
                        Fill = Brushes.Transparent,
                        PointGeometrySize = 5,
                        PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                    });
                }
                cartesianChart.AxisX.Add(new Axis
                {
                    Title = "Час",
                    Labels = dates,
                    //IsMerged = true,
                    Separator = new Separator
                    {
```

```
                        StrokeThickness = 1,
                        StrokeDashArray = new DoubleCollection(2),
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(64, 79, 86))

                    }
                });
                cartesianChart.AxisY.Add(new Axis
                {
                    IsMerged = true,
                    Separator = new Separator
                    {
                        StrokeThickness = 1,
                        StrokeDashArray = new DoubleCollection(4),
                        Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(64, 79, 86))
                    }
                });
                cartesianChart.Zoom = ZoomingOptions.X;
                TimeSpan interval = TimeSpan.FromMilliseconds(150);
                cartesianChart.AnimationsSpeed = interval;
            }
            else
            {
                VoltagSolarSeries = new LineSeries
                {
                    Values = ChartValues,
                    Name = "Напруга",
                    Title = "Напруга СП",
                    StrokeThickness = 5,
                    StrokeDashArray = new
System.Windows.Media.DoubleCollection(50),
                    Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(60, 179, 113)),
                    Fill = gradientBrush,
                    PointGeometrySize = 5
                };
                PowerSeries = new LineSeries
                {
                    Values = ChartValues1,
                    Name = "Потужність",
                    Title = "Потужність",
                    StrokeThickness = 5,
                    Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(123, 104, 238)),
                    Fill = Brushes.Transparent,
                    PointGeometrySize = 5,
                    PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                };
                AngleSeries = new LineSeries
                {
                    Values = ChartValues2,
                    Name = "Кут",
                    Title = "Кут",
                    StrokeThickness = 5,
                    Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(72, 61, 139)),
                    Fill = Brushes.Transparent,
                    PointGeometrySize = 5,
                    PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                };
                CurentSeries = new LineSeries
```

```
                {
                    Values = ChartValues3,
                    Name = "Струм",
                    Title = "Струм",
                    StrokeThickness = 5,
                    Stroke = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(255, 99, 71)),
                    Fill = Brushes.Transparent,
                    PointGeometrySize = 5,
                    PointForeground = new
SolidColorBrush(System.Windows.Media.Color.FromRgb(40, 26, 29))
                };
                cartesianChart.Series = new SeriesCollection
                    {
                        VoltagSolarSeries,
                        PowerSeries,
                        AngleSeries,
                        CurentSeries
                    };
                cartesianChart.AxisX.Add(new Axis
                {
                    DisableAnimations = true,
                    LabelFormatter = value => new
System.DateTime((long)value).ToString("mm:ss"),
                    Separator = new Separator
                    {
                        Step = TimeSpan.FromSeconds(1).Ticks
                    }
                });

                SetAxisLimits(System.DateTime.Now);
                Timer.Start();
            }

        }
        public LineSeries VoltagSolarSeries { get; set; }
        public LineSeries VoltagBatSeries { get; set; }
        public LineSeries PowerSeries { get; set; }
        public LineSeries AngleSeries { get; set; }
        public LineSeries CurentSeries { get; set; }
        public LineSeries Curent2Series { get; set; }
        public ChartValues<MeasureModel> ChartValues { get; set; }
        public ChartValues<MeasureModel> ChartValues1 { get; set; }
        public ChartValues<MeasureModel> ChartValues2 { get; set; }
        public ChartValues<MeasureModel> ChartValues3 { get; set; }

        public Timer Timer { get; set; }
        public Random R { get; set; }
        private void SetAxisLimits(System.DateTime now)
        {
            zoom = (Convert.ToByte(materialSlider_zoom.Value));
            cartesianChart.AxisX[0].MaxValue = now.Ticks +
TimeSpan.FromSeconds(1).Ticks;
            cartesianChart.AxisX[0].MinValue = now.Ticks -
TimeSpan.FromSeconds(zoom).Ticks;
        }
        private void TimerOnTick(object sender, EventArgs eventArgs)
        {
            var now = System.DateTime.Now;
            //test
            int volteg = R.Next(30, 33);
            int curent = R.Next(20, 22);
            int power = volteg * curent;
            int angel = 60;
```

```
            ChartValues.Add(new MeasureModel
            {
                DateTime = now,
                Value = volteg
            });
            ChartValues1.Add(new MeasureModel
            {
                DateTime = now,
                Value = power
            });
            ChartValues2.Add(new MeasureModel
            {
                DateTime = now,
                Value = angel
            });
            ChartValues3.Add(new MeasureModel
            {
                DateTime = now,
                Value = curent
            });
            SetAxisLimits(now);

            buffer_real_time =
Convert.ToInt32(materialSlider_size_buffer.Value);
            while (ChartValues.Count > buffer_real_time)
            {
                ChartValues.RemoveAt(0);
                ChartValues1.RemoveAt(0);
                ChartValues2.RemoveAt(0);
                ChartValues3.RemoveAt(0);
            }
        }
        private void button_open_file_Click(object sender, EventArgs e)
        {
            openFileDialog.InitialDirectory = (program_directory + @"\SPCS Solar
data");

            if (openFileDialog.ShowDialog() == DialogResult.Cancel)
            {
                return;
            }

            TextBox_file_data.Text = File.ReadAllText(openFileDialog.FileName);
        }
        private void Button_save_Click(object sender, EventArgs e)
        {
            if (saveFileDialog.ShowDialog() == DialogResult.Cancel)
            {
                return;
            }

            char[] arr_simv = { ':', ',', '@', '>', '<', '_', '\\', '|', '=',
'+', '-', '?', '/', '[', ']', '!' };
            string text = TextBox_file_data.Text;
            if (saveFileDialog.FileName.Contains("csv"))
            {
                for (int i = 0; i < arr_simv.Length; i++)
                {
                    if (text.Contains(arr_simv[i]))
                    {
                        text = text.Replace(arr_simv[i], ';');
                    }
```

```
                }
            }

            File.WriteAllText(saveFileDialog.FileName, text);
        }

        private void ComboBox_date_SelectedIndexChanged(object sender, EventArgs
e)
        {
            string selectedState = ComboBox_date.SelectedItem.ToString();
            switch (selectedState)
            {
                case "Час":
                    materialCombo_sort_time.Items.Clear();
                    materialCombo_sort_time.Items.AddRange(new string[] {
"Минутам", "Часам", "3 Часа", "Дням", "Неделям" });
                    materialCombo_sort_time.SelectedItem = "Минутам";
                    materialComboBox_sort_Y.Enabled = false;
                    materialCombo_sort_time.Enabled = false;
                    GrafUpdate(true, false);
                    break;
                case "День":
                    materialCombo_sort_time.Items.Clear();
                    materialCombo_sort_time.Items.AddRange(new string[] {
"Минутам", "Часам", "3 Часа" });
                    materialCombo_sort_time.SelectedItem = "Минутам";
                    materialCombo_sort_time.Enabled = true;
                    materialComboBox_sort_Y.Enabled = true;
                    GrafUpdate(true, false);
                    break;
                case "Неделю":
                    materialCombo_sort_time.Items.Clear();
                    materialCombo_sort_time.Items.AddRange(new string[] {
"Часам", "3 Часа", "Дням" });
                    materialCombo_sort_time.SelectedItem = "Часам";
                    materialCombo_sort_time.Enabled = true;
                    materialComboBox_sort_Y.Enabled = true;
                    GrafUpdate(true, false);
                    break;
                case "Месяц":
                    materialCombo_sort_time.Items.Clear();
                    materialCombo_sort_time.Items.AddRange(new string[] {
"Часам", "3 Часа", "Дням", "Неделям" });
                    materialCombo_sort_time.SelectedItem = "Часам";
                    materialCombo_sort_time.Enabled = true;
                    materialComboBox_sort_Y.Enabled = true;
                    GrafUpdate(true, false);
                    break;
                case "3 Месяца":
                    materialCombo_sort_time.Items.Clear();
                    materialCombo_sort_time.Items.AddRange(new string[] { "3
Часа", "Дням", "Неделям" });
                    materialCombo_sort_time.SelectedItem = "3 Часа";
                    materialCombo_sort_time.Enabled = true;
                    materialComboBox_sort_Y.Enabled = true;
                    GrafUpdate(true, false);
                    break;
            }
        }

        private void materialCheckbox_real_time_CheckedChanged(object sender,
EventArgs e)
        {
            if (materialCheckbox_real_time.Checked == true)
```

```
        {
            access = true;
            button_pause.Enabled = true;
            materialSlider_zoom.Enabled = true;
            materialSlider_size_buffer.Enabled = true;
            MaterialButtonUpdate.Enabled = false;
            GrafUpdate(true, true);
            CheckBoxRead();
        }
        else
        {
            access = false;
            button_pause.Enabled = false;
            button_pause.Text = "Пауза";
            materialSlider_zoom.Enabled = false;
            materialSlider_size_buffer.Enabled = false;
            MaterialButtonUpdate.Enabled = true;
            GrafUpdate(true, false);
        }
    }
    private void button_pause_Click(object sender, EventArgs e)
    {
        if (button_pause.Text == "Пауза")
        {
            Timer.Stop();
            button_pause.Text = "Відновити";
        }
        else
        {
            Timer.Start();
            button_pause.Text = "Пауза";
        }
    }
    void CheckBoxRead()
    {
        if (materialCheckbox_vs_graf_visible.Checked)
        {
            VoltagSolarSeries.Visibility = Visibility.Visible;
        }
        else
        {
            VoltagSolarSeries.Visibility = Visibility.Hidden;
        }
        if (materialCheckbox_p1_graf_visible.Checked)
        {
            PowerSeries.Visibility = Visibility.Visible;
        }
        else
        {
            PowerSeries.Visibility = Visibility.Hidden;
        }
        if (materialCheckbox_p2_graf_visible.Checked)
        {
            AngleSeries.Visibility = Visibility.Visible;
        }
        else
        {
            AngleSeries.Visibility = Visibility.Hidden;
        }
        if (materialCheckbox_c1_graf_visible.Checked)
        {
            CurentSeries.Visibility = Visibility.Visible;
        }
        else
```

```
            {
                CurentSeries.Visibility = Visibility.Hidden;
            }
        }
        private void materialCheckbox_vs_graf_visible_CheckedChanged(object
sender, EventArgs e)
        {
            if (access)
            {
                if (materialCheckbox_vs_graf_visible.Checked)
                {
                    VoltagSolarSeries.Visibility = Visibility.Visible;
                }
                else
                {
                    VoltagSolarSeries.Visibility = Visibility.Hidden;
                }
            }
        }
        private void materialCheckbox_p1_graf_visible_CheckedChanged(object
sender, EventArgs e)
        {
            if (access)
            {
                if (materialCheckbox_p1_graf_visible.Checked)
                {
                    PowerSeries.Visibility = Visibility.Visible;
                }
                else
                {
                    PowerSeries.Visibility = Visibility.Hidden;
                }
            }
        }
        private void materialCheckbox_p2_graf_visible_CheckedChanged(object
sender, EventArgs e)
        {
            if (access)
            {
                if (materialCheckbox_p2_graf_visible.Checked)
                {
                    AngleSeries.Visibility = Visibility.Visible;
                }
                else
                {
                    AngleSeries.Visibility = Visibility.Hidden;
                }
            }
        }
        private void materialCheckbox_c1_graf_visible_CheckedChanged(object
sender, EventArgs e)
        {
            if (access)
            {
                if (materialCheckbox_c1_graf_visible.Checked)
                {
                    CurentSeries.Visibility = Visibility.Visible;
                }
                else
                {
                    CurentSeries.Visibility = Visibility.Hidden;
                }
            }
        }
```

```
        private void MaterialButtonUpdate_Click(object sender, EventArgs e)
        {
            GrafUpdate(true, false);
        }
    }
    public class MeasureModel
    {
        public System.DateTime DateTime { get; set; }
        public double Value { get; set; }
    }
```

Форма FSettings

```
//////////////////////////////////////////////////////////////////////////////
//////

public partial class FormSetting : MaterialForm
    {
        public FormSetting()
        {
            InitializeComponent();
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.EnforceBackcolorOnAllComponents = true;
            materialSkinManager.AddFormToManage(this);
            ScanParametrs();
        }
        void ScanParametrs()
        {
            textBoxTime_Out.Text = Settings.Default["time_out"].ToString();
            TextBoxGetSource.Text = Settings.Default["GET_source"].ToString();
            checkBox_activ_log.Checked =
Convert.ToBoolean(Settings.Default["activ_log"]);
            materialCheck_dark_theme.Checked =
Convert.ToBoolean(Settings.Default["dark_theme"]);
            checkBox_enable_report_solar.Checked =
Convert.ToBoolean(Settings.Default["activ_report_solar"]);
            GET_interval_textBox.Text =
Settings.Default["polling_time"].ToString();
            textBox_volteg.Text =
Settings.Default["voltage_measurement_factor"].ToString();
            textBox_curent.Text =
Settings.Default["current_measurement_factor"].ToString();
            MaterialTextBoxAngle.Text =
Settings.Default["angle_measurement_factor"].ToString();
            textBox_data_period.Text =
Settings.Default["report_writing_frequency"].ToString();
            textBox_format_text.Text =
Settings.Default["type_path_in_file"].ToString();
            materialSlider1.Value =
Convert.ToInt32(Settings.Default["log_length"]);

            MaterialTextBoxPortMQTT.Text =
Settings.Default["mqtt_port"].ToString();
            MaterialTextBoxAdressBroker.Text =
Settings.Default["mqtt_ip"].ToString();
            MaterialTextBoxLogin.Text =
Settings.Default["mqtt_login"].ToString();
            MaterialTextBoxPassword.Text =
Settings.Default["mqtt_password"].ToString();
            MaterialCheckboxUseLocalBroker.Checked =
Convert.ToBoolean(Settings.Default["mqtt_broker"]);
        }
        void SelectThem(bool dark_theme)
```

```
        {
            var materialSkinManager = MaterialSkinManager.Instance;
            if (dark_theme)
            {
                materialSkinManager.Theme = MaterialSkinManager.Themes.DARK;
                materialSkinManager.ColorScheme = new ColorScheme(
                      materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal500 : Primary.Indigo500,
                      materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal700 : Primary.Indigo700,
                      materialSkinManager.Theme ==
MaterialSkinManager.Themes.DARK ? Primary.Teal200 : Primary.Indigo100,
                      Accent.Pink200,
                      TextShade.WHITE);
            }
            else
            {
                materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;
                materialSkinManager.ColorScheme = new
ColorScheme(Primary.Indigo500, Primary.Indigo700, Primary.Indigo100,
Accent.Pink200, TextShade.WHITE);
            }
        }
        ////////////////////////////////////////////////Текстовые
поля//////////////////////////////////////////////////
        private void GET_interval_textBox_TextChanged(object sender, EventArgs
e)
        {
            string temp = GET_interval_textBox.Text;
            if (temp != "")
            {
                Settings.Default["polling_time"] =
Convert.ToInt32(GET_interval_textBox.Text);
            }
        }
        private void textBox_volteg_TextChanged(object sender, EventArgs e)
        {
            string temp = textBox_volteg.Text;
            if (temp != "")
            {
                Settings.Default["angle_measurement_factor"] =
Convert.ToDouble(textBox_volteg.Text);
            }
        }
        private void textBox_curent_TextChanged(object sender, EventArgs e)
        {
            string temp = textBox_curent.Text;
            if (temp != "")
            {
                Settings.Default["current_measurement_factor"] =
Convert.ToDouble(textBox_curent.Text);
            }
        }
        private void textBox_format_text_TextChanged(object sender, EventArgs e)
        {
            Settings.Default["type_path_in_file"] = textBox_format_text.Text;
        }
        private void textBox_data_period_Leave(object sender, EventArgs e)
        {
            string temp = textBox_data_period.Text;
            if (temp != "")
            {
                Settings.Default["report_writing_frequency"] =
Convert.ToInt16(textBox_data_period.Text);
```

```
            }
        }
        private void TextBoxGetSource_Leave(object sender, EventArgs e)
        {
            Settings.Default["GET_source"] = TextBoxGetSource.Text;
        }
        private void MaterialTextBoxPortMQTT_Leave(object sender, EventArgs e)
        {
            if(MaterialTextBoxPortMQTT.Text != "")
            {
                Settings.Default["mqtt_port"] = MaterialTextBoxPortMQTT.Text;
            }
        }
        private void MaterialTextBoxAdressBroker_Leave(object sender, EventArgs
e)
        {
            if (MaterialTextBoxAdressBroker.Text != "")
            {
                Settings.Default["mqtt_ip"] = MaterialTextBoxAdressBroker.Text;
            }
        }
        private void MaterialTextBoxLogin_Leave(object sender, EventArgs e)
        {
            if (MaterialTextBoxLogin.Text != "")
            {
                Settings.Default["mqtt_login"] = MaterialTextBoxLogin.Text;
            }
        }
        private void MaterialTextBoxPassword_Leave(object sender, EventArgs e)
        {
            if (MaterialTextBoxPassword.Text != "")
            {
                Settings.Default["mqtt_password"] =
MaterialTextBoxPassword.Text;
            }
        }
        private void MaterialTextBoxAngle_Leave(object sender, EventArgs e)
        {
            string temp = MaterialTextBoxAngle.Text;
            if (temp != "")
            {
                Settings.Default["voltage_measurement_factor"] =
Convert.ToDouble(textBox_volteg.Text);
            }
        }

        ////////////////////////////////////////////////Чек
бокс//////////////////////////////////////////////////
        private void checkBox_activ_log_CheckedChanged(object sender, EventArgs
e)
        {
            Settings.Default["activ_log"] = checkBox_activ_log.Checked;
            Settings.Default.Save();
        }
        private void checkBox_dark_theme_CheckedChanged(object sender, EventArgs
e)
        {
            bool dark_theme = materialCheck_dark_theme.Checked;
            Settings.Default["dark_theme"] = dark_theme;
            Program.FM.SetGrafSolid(dark_theme);
            SelectThem(dark_theme);
        }
        private void textBox2_TextChanged(object sender, EventArgs e)
        {
```

```
        string temp = textBoxTime_Out.Text;
        if (temp != "")
        {
            Settings.Default["time_out"] = Convert.ToInt64(temp);
            Settings.Default.Save();
        }
    }
    private void KeyPresGET_interval(object sender, KeyPressEventArgs e)
    {
        char number = e.KeyChar;
        if (!Char.IsDigit(number) && number != 8)
        {
            e.Handled = true;
        }
    }
    private void KeyPresFactor(object sender, KeyPressEventArgs e)
    {
        char number = e.KeyChar;
        if ((Convert.ToString(textBox_volteg.Text)).IndexOf(",") == -1)
        {
            if (!Char.IsDigit(number) && number != 8 && number != 44)
            {
                e.Handled = true;
            }
        }
        else
        {
            if (!Char.IsDigit(number) && number != 8)
            {
                e.Handled = true;
            }
        }
    }
    private void buttonreset_Click(object sender, EventArgs e)
    {
        Settings.Default.Reset();
        ScanParametrs();
    }
    private void form_close(object sender, FormClosedEventArgs e)
    {
        Settings.Default.Save();
    }

    ////////
    private void KeyPresFactorC(object sender, KeyPressEventArgs e)
    {
        char number = e.KeyChar;
        if ((Convert.ToString(textBox_curent.Text)).IndexOf(",") == -1)
        {
            if (!Char.IsDigit(number) && number != 8 && number != 44)
            {
                e.Handled = true;
            }
        }
        else
        {
            if (!Char.IsDigit(number) && number != 8)
            {
                e.Handled = true;
            }
        }
    }

    private void folderBrowserDialog_HelpRequest(object sender, EventArgs e)
```

```
        {
            FolderBrowserDialog browserDialog = new FolderBrowserDialog();
            if (browserDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
            {
                Settings.Default["data_file_path"] = browserDialog.SelectedPath;
            }
        }

        private void button_cur_1_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{CURRENT}");
        }

        private void button_cur_2_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{ANGLE}");
        }

        private void button_vol_1_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{VOLTAGE}");
        }

        private void button_pow_1_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{POWER}");
        }

        private void button_pow_all_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{WORK_MOD}");
        }

        private void button_time_Click(object sender, EventArgs e)
        {
            string temp;
            temp = textBox_format_text.Text;
            textBox_format_text.Text = (temp + "{TIME}");
        }

        private void checkBox_enable_report_solar_CheckedChanged(object sender,
EventArgs e)
        {
            Settings.Default["activ_report_solar"] = checkBox_activ_log.Checked;
        }

        private void materialSlider1_Click(object sender, EventArgs e)
        {
            Settings.Default["log_length"] =
Convert.ToInt32(materialSlider1.Value);
        }
```

```
        private void MaterialButtonVidPidConfigurator_Click(object sender,
EventArgs e)
        {
            MaterialDialog materialDialog = new MaterialDialog(this, "Увага!",
"Відключіть модуль від програми, якщо він підключений.", "OK", true,
"Скасувати");
            DialogResult result = materialDialog.ShowDialog(this);
            if (result.ToString() == "OK")
            {
                System.Diagnostics.Process.Start(@".\MCP_VID_PID.exe");
            }
            materialDialog.Close();
        }

        private void MaterialCheckboxUseLocalBroker_CheckedChanged(object
sender, EventArgs e)
        {
            if (MaterialCheckboxUseLocalBroker.Checked == true)
            {
                MaterialDialog materialDialog = new MaterialDialog(this,
"Увага!", "Для запуску брокера необхідно перезапустити програму!", "OK", true,
"Скасувати");
                DialogResult result = materialDialog.ShowDialog(this);
                if (result.ToString() == "OK")
                {
                    MaterialTextBoxPortMQTT.Text = "1883";
                    MaterialTextBoxPortMQTT.Enabled = false;
                    MaterialTextBoxAdressBroker.Text = "localhost";
                    MaterialTextBoxAdressBroker.Enabled = false;
                    MaterialTextBoxLogin.Text = "SPCS";
                    MaterialTextBoxLogin.Enabled = false;
                    MaterialTextBoxPassword.Text = "3676";
                    MaterialTextBoxPassword.Enabled = false;

                    Settings.Default["mqtt_login"] = MaterialTextBoxLogin.Text;
                    Settings.Default["mqtt_ip"] =
MaterialTextBoxAdressBroker.Text;
                    Settings.Default["mqtt_password"] =
MaterialTextBoxPassword.Text;
                    Settings.Default["mqtt_port"] =
MaterialTextBoxPortMQTT.Text;
                    Settings.Default["mqtt_broker"] =
MaterialCheckboxUseLocalBroker.Checked;
                }
                else
                {
                    MaterialCheckboxUseLocalBroker.Checked = false;
                }
                materialDialog.Close();
            }
            else
            {
                MaterialTextBoxPortMQTT.Text = "";
                MaterialTextBoxPortMQTT.Enabled = true;
                MaterialTextBoxAdressBroker.Text = "";
                MaterialTextBoxAdressBroker.Enabled = true;
                MaterialTextBoxLogin.Text = "";
                MaterialTextBoxLogin.Enabled = true;
                MaterialTextBoxPassword.Text = "";
                MaterialTextBoxPassword.Enabled = true;

                Settings.Default["mqtt_broker"] =
MaterialCheckboxUseLocalBroker.Checked;
            }
```

```
        }
    }
```

Форма FormUpdt

```
//////////////////////////////////////////////////////////////////////////////
//////

public partial class FormUpdt : MaterialForm
    {
        readonly WebClient UP_client = new WebClient();
        readonly FileManager fileManager = new FileManager();
        string curen_version;
        readonly Uri source_up = new
Uri("https://drive.google.com/uc?export=download&confirm=no_antivirus&id=1OUESR6
b6_qmM8K_xMHoqcg1ZyHcCisXb");

        public FormUpdt()
        {
            InitializeComponent();
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.EnforceBackcolorOnAllComponents = true;
            materialSkinManager.AddFormToManage(this);
            curen_version = Settings.Default["curen_version"].ToString();
            labelactual_ver.Text = curen_version;
        }

        private void button_scan_upd_Click(object sender, EventArgs e)
        {
            try
            {
                string new_version =
UP_client.DownloadString("https://pastebin.com/2xuDVvxK");
                //Начало и конец прочитанного файла.
                int version_start = new_version.LastIndexOf("<textarea
class=\"textarea -raw js-paste-raw\">") + 45;
                int version_end = new_version.IndexOf("</textarea>");

                label_new_version.Text = new_version.Substring(version_start,
(version_end - version_start)); // Вывод актуальной версии.

                if (new_version.Contains(curen_version))
                {
                    MessageBox.Show("Установлена последняя версия.",
"Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    MessageBox.Show("Доступно обновление.", "Уведомление",
MessageBoxButtons.OK, MessageBoxIcon.Warning);

                    //Включение скрытых обьектов, отключение кнопки проверки
обновления.
                    label_new_ver_st.Visible = true;
                    label_new_version.Visible = true;
                    button_setup.Visible = true;
                    progressBar_setup.Visible = true;
                    button_scan_upd.Enabled = false;
                }
            }
            catch
            {
                MessageBox.Show("При попытке подключения возникла ошибка.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }
        }
        private void DownloadFileCallback(object sender, AsyncCompletedEventArgs
e)
        {
            if (e.Cancelled == false)
            {
                MessageBox.Show("Файл обновления загружен.", "Уведомление",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                string zipPath = @".\UPDATE.zip";
                string extractPath = @".\Update";
                try
                {
                    ZipFile.ExtractToDirectory(zipPath, extractPath);
                    System.Diagnostics.Process.Start(@"Update\SPCS.exe");
                    File.AppendAllText("Data SPCS\\data_up.txt", "1");
                    this.Close();
                }
                catch
                {

                }
            }
            if (e.Error != null)
            {
                MessageBox.Show("Ошибка!", "Уведомление", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        void client_DownloadProgressChanged(object sender,
DownloadProgressChangedEventArgs e)
        {
            this.BeginInvoke((MethodInvoker)delegate
            {
                double bytesIn = double.Parse(e.BytesReceived.ToString());
                double totalBytes =
double.Parse(e.TotalBytesToReceive.ToString());
                double percentage = bytesIn / totalBytes * 100;
                progressBar_setup.Value =
int.Parse(Math.Truncate(percentage).ToString());
            });
        }

        private void button_setup_Click(object sender, EventArgs e)
        {
            try
            {
                button_setup.Enabled = false;
                UP_client.DownloadFileCompleted += new
AsyncCompletedEventHandler(DownloadFileCallback);
                UP_client.DownloadProgressChanged += new
DownloadProgressChangedEventHandler(client_DownloadProgressChanged);
                UP_client.DownloadFileAsync(source_up, "UPDATE.zip");
            }
            catch
            {
                button_setup.Enabled = true;
            }
        }
    }
```

ВІДГУК КОНСУЛЬТАНТІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ВІДГУК

на тему «Автоматизація процесів керування сонячною фотоелектричною установкою», виконаної студентом групи 151-18-1 Рульовим Є.І.

Мета роботи полягає в забезпеченні енергоефективності використання сонячних фотоелектричних перетворювачів (далі - СФП). Проведено аналіз причин неможливості роботи СФП з повною віддачею протягом усієї світлої пори доби.

На підставі результатів проведеної роботи створив алгоритм керування СФП з ідентифікацією системою керування факторів, які наразі впливають на генеровану потужність. Розроблений алгоритм дозволить підвищити ККД за рахунок розпізнавання різних ситуацій при роботі та забезпечить оптимальне положення СФП у просторі в залежності від конкретної ситуації.

Основні теоретичні результати кваліфікаційної роботи пройшли перевірку на експериментальній установці.

За час виконання кваліфікаційної роботи студент показав високі теоретичні знання та практичні навички. Кваліфікаційна робота виконана самостійно, в повному обсязі у відповідності до чинних вимог.

Вважаю, що кваліфікаційна робота заслуговує оцінки відмінно (90 балів), а студент Рульов Є.І. присвоєння кваліфікації бакалавр за спеціальністю «151 Автоматизація та комп'ютерно-інтегровані технології».

Керівник кваліфікаційної роботи

(посада, вчене звання, ступінь) (підпис) (ініціали, прізвище)

РЕЦЕНЗІЯ

на тему «Автоматизація процесів керування сонячною фотоелектричною установкою», виконаної студентом групи 151-18-1 Рульовим Є.І.

Завдання і зміст кваліфікаційної роботи ступеню бакалавра відповідає основній меті – перевірці знань та ступеню підготовки здобувача вищої освіти за спеціальністю “151 Автоматизація та комп’ютерно-інтегровані технології”. Оформлення пояснювальної записки та графічних матеріалів кваліфікаційної роботи виконано відповідно до вимог стандартів та методичних рекомендацій повністю.

Актуальність роботи полягає в тому, що розробка нової системи керування дозволить зменшити економічні витрати та збільшити прибутковість від продажу енергії.

Повнота та глибина вирішення поставлених завдань в кваліфікаційній роботі достатня.

В рамках кваліфікаційної роботи виконано аналіз технологічного процесу та об’єкта керування, постановка завдання, вибір апаратного забезпечення, розробка програмного забезпечення, розрахунок основних економічних показників та вирішення питань з охорони праці.

В цілому кваліфікаційна робота ступеню бакалавра заслуговує оцінки “90” балів при відповідному захисті, а здобувач Рульов Є.І. присвоєння кваліфікації “бакалавр” за спеціальністю “151 Автоматизація та комп’ютерно-інтегровані технології”.

Рецензент, _____

___.06.2022