МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні

Кафедра електроніки, інформаційних систем
та програмного забезпечення
(повна назва кафедри)

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: Дослідження та розробка автономної мінітеплиці з дистанційним контролем параметрів

Виконав: студент II курсу, групи 8.1531з
спеціальності 153 «Мікро- та наносистемна техніка
(код і назва спеціальності)
освітньої програми Мікроелектронні інформаційні системи
(код і назва освітньої програми)
Жарков Євген Вячеславович
(ініціали та прізвище)

Керівник доцент кафедри ЕІСПЗ, доцент, к.т.н.
Ніконова Аліна Олександрівна
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент генеральний директор ТОВ «Омега, ЛТД»

Шевченко Тамара Василівна
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні

Рівень вищої освіти ___*другий магістерський*_____
(перший (бакалаврський) рівень
Напрям підготовки _____*153*_____
(шифр)

Спеціальність __*153 Мікро- та наносистемна техніка*_____
(шифр)

ЗАТВЕРДЖУЮ
Завідувач кафедри___*ЕІСПЗ*_____
Т.В.Критська_____
«___»_____2022року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВУЧУ ВИЩОЇ ОСВІТИ

_____*Жарков Євген Вячеславович*_____
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи магістра *Дослідження та розробка автономної мінітеплиці з дистанційним контролем параметрів*_____

керівник кваліфікаційної роботи бакалавра *Ніконова Аліна Олександрівна*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "02"червня 2022 року №598-с

2. Строк подання студентом кваліфікаційної роботи 01 грудня 2022 року

3. Вихідні дані до кваліфікаційної роботи _мікроконтролет ATmega328P-PU, датчик температури та вологості АНТ10, перетворювач інтерфейсу pcf8674 стабілізатор напруги l7805cv, зовнішній тактовий генератора побудований на кварцевому резонаторі

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____

1 Дослідження особливостей конструкцій систем автоматичного вирощування рослин

2 Розробка системи автоматичного вирощування рослин

3 Техніко-економічне обгрунтування

4 Охорона праці та техногенна безпека

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Л1- ПСистема автоматичного вирощування рослин; Л2- Переваги мікроконтролеру Atmega328; Л3 – Обгрунтування вибору датчиків контролю параметрів; Л4-Блок комутації; Л5-моделювання схеми з дикористанням двунаправленої шини передачи даних; Л6-Електрична схема приладу; Л7-Топологія друкованої плати; Л8- Розробка прототипу приладу

6. Консультанти розділів кваліфікаційної роботи бакалавра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання прийняв
I	*Ніконова А.О.*	*02.06.2022*
II	*Ніконова А.О.*	*20.09.2022*
III	*Ніконова А.О.*	*12.10.2022*
IV	*Ніконова А.О.*	*15.11.2022*

7. Дата видачі завдання *02.06.2022*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів кваліфікаційної роботи бакалавра	Примітка
1	*Огляд існуючих Гроубоксів*	*02.06-30.06*	
2	*Визначення основних параметрів мікроклімату*	*30.08-10.09*	
3	*Розробка структурної схеми*	*10.09-28.09*	
4	*Підключення датчиків температури та вологості*	*28.09-12.10*	
5	*Розробка блоку комутації*	*12.10-28.10*	
6	*Розробка схеми блоку живлення*	*28.10-02.11*	
7	*Моделювання схеми в електронному середовищі*	*02.11-12.11*	
8	*Розділ економічного обгрунтування*		
9	*Розділ охорони праці та техногенної безпеки*	*12.11-16.11*	
10	*Оформлення плакатів*	*16.11-25.11*	
11	*Доклад*	*25.11-01.12*	

Студент _____ *Жарков Євген Вячеславович*
(підпис) (прізвище та ініціали)

Керівник роботи (проекту) _____ *Ніконова Аліна Олександрівна*
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ *Верьовкін Леонід Леонідович*
(підпис) (прізвище та ініціали)

Реферат

Дипломна робота містить 118 сторінку, 33 рисунки, 9 таблиць, 16 джерела літератури.

Об’єкт дослідження – автономна мінітеплиця з дистанційним контролем параметрів.

Мета роботи – розробка автономної мінітеплиці з дистанційним контролем параметрів та керуванням

Завдання роботи – забезпечити точність передачі інформації, синхронність функціонування вузлів приладу; забезпечити контроль параметрів за допомогою датчиків, мобільність і автономність використання мінітеплиці.

Методика досліджень – моделювання пристрою за допомогою програмних забезпечень Proteus, SPlan 5.0, Layout 4.0.

Короткий виклад результатів досліджень – за допомогою розробленої системи можна керувати насосом, освітленням, обігрівачем, вентилятором, зволожувачем повітря. Таким чином можна організувати автополив, включати і відключати освітлення в потрібний час, забезпечити роботу вентиляції за таймером або при високій температурі, контролювати температуру обігрівача при різних температурних режимах.

Результати впроваджень – макет електронного приладу пройшов випробовування на кафедрі ЕІСПЗ.

Прогнозні пропозиції – рекомендується подальша доробка схеми, для забезпечення запам’ятовування інформації і передачіі її на персональний комп’ютер або мобільний пристрій для подальшого аналізу.

МІКРОКОНТРОЛЕР, ГРОУБОКС, ДАТЧИК, ШИНА ПЕРЕДАЧІ ДАНИХ, ІНДИКАТОР, КОМУТАТОР, РЕЗИСТОР, КОНДЕНСАТОР

Дипломну роботу виконано на кафедрі електроніки, інформаційних систем та програмного забезпечення з 01.09.2022 р. по 10.12.2022 р.

ЗМІСТ

ВСТУП

У сільськогосподарському виробництві зайнято близько половини економічно активного населення світу. Сільське господарство - одна з провідних галузей економіки України. Світовий внесок нашої країни у виробництво конкретних видів зерна розрахувало Міністерство сільського господарства США. За його даними, в минулому маркетинговому році Україна посіла друге місце у світі за обсягом поставок ячменю, четверте – за експортом кукурудзи і п'яте – з продажу на світових ринках пшениці.

Для забезпечення нормальних умов росту рослин, наближених до оптимальних, найбільш перспективним є створення системи контролю і регулювання із зворотнім зв’язком (так звані слідкуючі системи). Електронні системи в даному випадку найбільш прийнятні: вони мають високу гнучкість контролю параметрів середовища у великих діапазонах, можливість створення компактних і дешевих функціональних блоків і вузлів, практично не мають інерції, не потребують будь-якого дорогого обладнання і пристроїв для їх створення.

Використання в замкнутих об’ємах електронних систем забезпечення мікроклімату економить час та ресурси їх власників, а автоматний режим роботи допускає довгострокову відсутність власників. Така система дозволяє як мінімум забезпечувати контроль і регулювання наступних основних параметрів мікроклімату.

З огляду на вищезазначене, в Україні застосування автоматизованих систем управління мікрокліматом у теплицях є перспективною областю. Створення актуальних товарів для сільськогосподарського ринку позитивно позначиться на імпортозаміщенні та розвитку країни. Автоматизовані системи управління мікрокліматом з нижчою вартістю та простотою в експлуатації знизить вартість та утримання самих теплиць.

1 АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ СИСТЕМ АВТОМАТИЧНОГО ЗРОШУВАННЯ

Гроубокси в наші дні стають все популярнішими в домашніх умовах. Основним виробником є зарубіжні компанії. Серед Гроубоксів є відмінності за функціоналом та методом експлуатації. Кожна компанія прагне зробити свій продукт оригінальним, і тому з'являються додаткові функції, оригінальне дизайнерське рішення та практичне використання. На даний момент на ринку фітотронів або інакше Grow box займають зарубіжні виробники, але український ринок не стоїть на місці та рухається вперед. На даний момент є одна компанія, яка займається виготовленням засобів засобів захисту рослин. А саме ALFA Smart Agro збудувала у Білій Церкві новий сучасний завод по виготовленню засобів захисту рослин.

1.1 Порівняльнй аналіз існуючих Гроубоксів

Для виявлення позитивних та негативних факторів, проведемо огляд моделей провідних виробників Гроубоксів за функціоналом та зручністю використання.

Urban Cultivator Inc є автоматизованим критим приладом для вирощування трав, мікроелементів, овочів і квітів.

Фітотрон Urban Cultivator Residential простий у використанні, з автоматичними функціями. Дозволяє виростити 8 сортів трав та мікрозелені одразу. Можливий полив водопровідною водою.

Рисунок 1.1 - Фітотрон Urban Cultivator Residential

Основні технічні характеристики фітотрону:

- Напруга живлення 110В або 220В
- Середньомісячні кВт: 18 кВт
- Автоматичний полив

Виробник Mountain View, Каліфорнія, США [1]

Grow Box - це зрошувальна система та клімат-контролер.

Grow Box має вбудований резервуар, водяний насос та датчик вологості для води. Система поливу спроектована таким чином, щоб забезпечити оптимальний та рівномірний полив, а балон із вбудованим індикатором дозволяє лише кілька разів на рік замінювати воду.

З огляду наведених моделей, які представлені на ринку аналоги Гроубоксів мають схожі особливості, більшість з них пророщують мікрозелень на гідропоніці, що задає певну конструкцію, резервуар з водою, насоси та трубки поливу, так само джерелом освітлення є світлодіоди, переважно червоного та

синього кольору. Для візуального полегшення конструкції та естетичного вигляду використовують такі матеріали, як: скло та метал. Кожен аналог має свої особливості: створення розумного тонування, розробка програми для гаджетів і т.д.

1.2 Визначення складових автоматизованої системы зрошування

Автоматизація теплиці має на увазі під собою відстеження різних показників та управління мікрокліматом для росту рослин.

Розумна теплиця здатна контролювати:

• тепло – запобігання перегріву або замерзанню рослин;

• воду – оскільки до теплиці не потрапляють опади, необхідно керувати поливом рослин;

• світло – додаткове підсвічування рослин чи їх затемнення;

• витратою повітря та вологості – щільно закрита теплиця приведе до підвищення вологості та нестачі кисню та вуглекислого газу для рослин залежно від часу доби;

• комах – можна не допустити проникнення шкідливих комах у теплицю або забезпечити комфортні умови для проживання корисних. Для кращого росту рослин необхідно одночасно контролювати більшу кількість цих показників . Це можуть забезпечити такі системи: • зрошення – регулярне надходження води за певним графіку;

• вентиляцію – увімкнення або вимкнення вентиляторів, або відкривання та закривання кватирок;

• дозування поживних речовин – за допомогою аналізу ґрунту можна розподіляти поживні речовини за системою зрошення;

• боротьба зі шкідниками – автоматичне обприскування рослин. Для найбільшої автоматизації та регулювання мікроклімату всі ці системи повинні керувати

одночасно і становити одну велику систему, яка зможе оптимізувати їхню роботу.

Підтримання заданих кліматичних параметрів є невід'ємною частиною нормального функціонування системи мікроклімату. Підбір оптимальних, близьких до ідеальних умов зростання, в даній роботі, для рослин є важливою частиною, адже на них ґрунтується мікроклімат [4,5]. Основні завдання системи автоматичного регулювання полягають в:

- керування температурою повітря;
- управління системою поливу;
- управління освітлювальними установками. Раніше автоматизація роботи теплиці була дорогою, а часом і не процедури, що окупається, але на даний момент вирішення цієї проблеми не настільки дорого і цілком окупається, а надалі, приносить ще більшу вигоду Мікроконтролери - це програмована мікросхема, яка що дозволяє здійснювати управління різними електронними пристроями. Мікроконтролер містить одне або кілька процесорних ядер, пам'ять, а також програмовані периферійні пристрої введення та виведення.

1.3 Реалізація систем управління

Більшість сучасних цифрових систем управління реалізуються на базі мікроконтролерів (МК) – спеціалізованих мікропроцесорних пристроїв, орієнтованих на виконання керуючих функцій. Інтегруючи на одному кристалі високопродуктивний процесор, пам'ять і набір периферійних пристроїв, мікроконтролери дозволяють з мінімальними витратами реалізувати високоефективні системи і пристрої управління. В даний час однокристальні мікроконтролери є найбільш масовими представниками мікропроцесорної техніки, обсяг випуску яких становить близько 2,5 млрд. штук на рік. Їх виробництвом займається ряд фірм (Intel, Motorola, Philips, Siemens, Atmel, Dallas, Temic, Oki, AMD і ін.), які

надають споживачам більшу номенклатуру виробів, починаючи з простих 8-розрядних моделей і закінчуючи 32-розрядними пристроями, ядром яких служать CISC-iRISC-процесори.

Розрізняють наступні типи мікроконтролерів:

1) Периферійні (інтерфейсні) МК призначені для реалізації найпростіших МП систем управління. Вони мають малу продуктивність і малі габаритні розміри. Зокрема можуть використовуватися периферійними пристроями ЕОМ (клавіатура, миша і т.п.). До них відносяться: AVR-Atmel, PIC – Micro Chip, VPS – 42 (Intel).

2) Універсальні 8-розрядні МК призначені для реалізації МП систем малої і середньої продуктивності. Мають просту систему команд і велику номенклатуру вбудованих пристроїв. Основні типи: MCS – 51 (Intel), Motorola HC05 – HC012 та 96н.

3) Універсальні 16-розрядні МК призначені для реалізації систем реального часу середньої продуктивності. Структура і система команд адаптовані на швидку реакцію за зовнішніми подіями. Найбільше використання мають в системах управління електродвигунами. До типових 16-розрядних МК відносяться: MCS96/196/296 (Intel), C161-C167 (Siemens, Infineon), HC16 Motorola та 96н.

4) Спеціалізовані 32-розрядні МК реалізують високопродуктивну архітектуру і призначені для систем телефонії, передачі інформації, телебачення і інших, що вимагають високошвидкісної обробки інформації.

5) Цифрові сигнальні процесори (DSP – Digital Signal Processor) призначені для складної математичної обробки вимірюваних сигналів у режимі реального часу. Широко використовуються в телефонії та зв'язку.

Основні відмінності DSP: підвищена розрядність оброблюваних слів (16,32,64 біта) і висока швидкість у форматі з плаваючою точкою (16 flops). Виробники: Texas Instruments (TMS 320 та 97н.), Analog Device (ADSP 2181 і 97н.). [2]

Розглянемо основні типи архітектури та системи команд процесорів МК.

У сучасних МК використовуються такі типи системи команд процесорів:

- RISC – (Reduce Instruction Set Commands) архітектура зі скороченим набором команд.
- CISC – (Complex Instruction Set Commands) традиційна архітектура з розширеним набором команд.
- ARM – (Advanced RISC – machine) вдосконалена RISC архітектура.

Головне завдання RISC архітектури забезпечення найвищої продуктивності процесора. Її характерними ознаками є:

- Мала кількість команд процесора (кілька десятків);
- Кожна команда виконується за мінімальний час (1-2 машинних цикла, такта).
- Максимально можлива кількість регістрів загального призначення процесора (кілька тисяч);
- Збільшена розрядність процесора (12,14,16 біт).

Сучасна RISC архітектура включає, як правило, тільки останні 3 пункти, тому що за рахунок щільності компонування ВІС стало можливим реалізувати велику кількість команд. У сучасних 32-розрядних МК використовують ARM архітектуру

1.4 Датчики контролю мікроклімату

Для забезпечення сприятливих умов росту рослин, так само необхідно здійснювати контроль за довкіллям. Як датчик довкілля використовується датчик DHT-11. Датчик DHT11 складається з двох частин з гідрометра та ємнісного датчика температури. Гігрометр вимірює вологість повітря. Також у датчику присутній контролер, який виконує аналого-цифрові перетворення для передачі цифрового сигналу на мікроконтролер.

• Точність вимірювання вологості ±3 %RH (у діапазоні 20…80 %RH)

• Точність вимірювання тиску ±1.0 hPa (в діапазоні 300 . . . 1100 hPa)

• Точність вимірювання температури ±0.5 °C (в діапазоні -40…+85 °C) Модуль розрахований на використання у різних мобільних пристроях та проектах. Тому він займає трохи місця та витрачає мало енергії. Зовнішній вигляд датчика зображено рисунку 2.1.

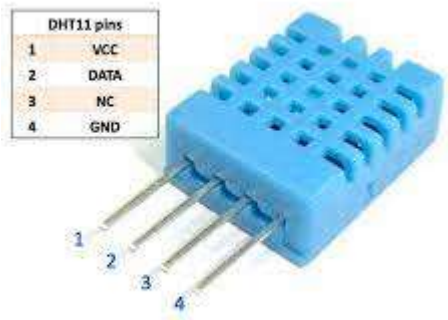


Рисунок 1.2 - датчик довкілля

Для прив'язки системи автоматичного поливу, крім встановленого таймера на включення, до вологості ґрунту необхідно використовувати гігрометр, датчик вологості ґрунту.

Модуль складається із двох частин:

• контактного щупа YL-69

• датчика YL-38 У комплекті йдуть дроти для підключення. Між двома електродами щупа YL-69 створюється невелика напруга. Якщо ґрунт сухий, опір великий і струм буде меншим. Якщо земля волога опір менший, струм - трохи більше. За підсумковим аналоговим сигналу можна судити про рівень вологості. Щуп YL-69 з'єднаний з датчиком YL-38 по двох дротах. Крім контактів з'єднання із щупом, датчик YL-38 має чотири контакти для підключення до контролера. Датчик YL-38 зібраний на основі компаратора LM393, що видає напруга на вихід D0 за принципом: вологий ґрунт – низький логічний рівень, сухий ґрунт – високий логічний рівень. Рівень визначається граничним значенням, яке можна регулювати за допомогою потенціометра. На висновок A0 подається аналогове значення, яке можна передавати в контролер для подальшої обробки, аналізу та

прийняття рішень. Датчик YL-38 має два світлодіоди, що сигналізують про наявність що надходить на датчик живлення та рівня цифрового сигналу на виході D0. Наявність цифрового виведення D0 та світлодіода рівня D0 дозволяє використовувати модуль автономний, без підключення до контролера.

Технічні характеристики:

- Напруга живлення – 3.3-5 В
- Струм споживання – 35 мА
- Вихід - цифровий та аналоговий
- Розмір модуля – 16×30 мм
- Розмір щупа – 20×60 мм
- Загальна вага – 7.5 г

У теплиці, що розробляється, використовується дугова натрієва лампа трубчастого типу, ДНаТ, робоча температура якої становить близько 350С, у зв'язку з чим необхідно передбачити систему визначення пожежі, а також наявності горючих, легкозаймистих газів. Спеціально для виявлення отруйних, горючих газів, а також зважених частинок, що є результатом горіння існує ціла серія датчиків MQ. У таблиці 1.1 представлені деякі з них

Таблиця 1.1 – Датчики серії MQ та гази, які вони визначають

Датчик	Визначення газу
MQ-2	Широкого спектру газів (пропан, бутан, метан та водень)
MQ-3	Пари спирту
MQ-4	Природний газ (CH4)
MQ-5	Горючі гази
MQ-6	Зріджені нафтові гази
MQ-7	Чадний газ
MQ-9	Горючі та чадні гази
MQ-131	Озон (O3)

У зв'язку з тим, що ми вибираємо датчик для визначення виникнення пожежі автоматизовану теплицю, оптимальним варіантом послужить датчик MQ-2 (рис.1.2) . Він визначає концентрацію вуглеводневих газів (пропан, метан, н-бутан), диму (зважених частинок, що є результатом горіння) та воднюу навколишньому середовищі. Датчик можна використовувати для виявлення витоків газу та задимлення.

Рисунок 1.2 - Датчик MQ-2

Датчики контролю системи автоматичного поливу, а також виконавчі механізми системи автоматичного керування. У процесі моделювання автоматизованої теплиці було виявлено необхідність використання датчика рівня води. Даних датчика необхідно два, так як один датчик є вимірником рівня води резервуар системи автоматичного поливу, другий датчик вимірює рівень води в піддоні, для включення відкачувального насоса з піддону до основного резервуар. Датчик рівня води призначений для визначення рівня води в різних ємностях, де недоступний візуальний контроль, з метою попередження переповнення ємності водою через критичну позначку.

Конструкції датчиків рівня води можуть бути різними: поплавкові, занурені, врізні. Даний датчик води – занурений. Чим більше занурення датчика у воду, тим менший опір між двома сусідніми проводами.

На контакт подається аналогове значення, яке можна передавати в контролер для подальшої обробки, аналізу та прийняття рішень. Датчик має червоний світлодіод, що сигналізує про наявність що надходить на датчик живлення (рис 1.3).

Розглянемо технічні характеристики даного датчика рівня води:

- Напруга живлення - 3.3-5 В
- Струм споживання - 20 мА
- Вихід – аналоговий
- Зона виявлення – 16×30 мм
- Розміри – 62×20×8 мм
- Робоча температура – 10 – 30 °C

Оскільки форма зони виявлення обмежена, так само з'являється обмеження глибини використовуваних резервуарів. Слід зазначити, що визначення точного рівня води в резервуарі системи автоматичного поливу не є обов'язковою умовою при монтажі та підборі деталей для поливу.

Достатньо буде розмістити його у верхній частині резервуара, для визначення переповнення, проте таке можливе лише за стороннього додаванні води в піддон, звідки відкачується вода в резервуар системи поливу при повідомленні датчиком рівня води в піддоні про необхідність включення відкачувального насоса.

Рисунок 1.3 – Датчик рівня води

Для створення системи автоматичного поливу необхідно створювати тиск у системі подачі води, та використовувати розпилювач. Для рівномірного зрошення грунту, необхідно використання, кількох розпилювачів, залежно від площі поверхні ґрунту. У проектованій системі через використовуються розміри, достатньо пари розпилювачів типу “Крапельниці” Розпилювач та клапан зображені на рисунках 1.4, 1.5 відповідно.

Рисунок 1.4 – Розпилювач води

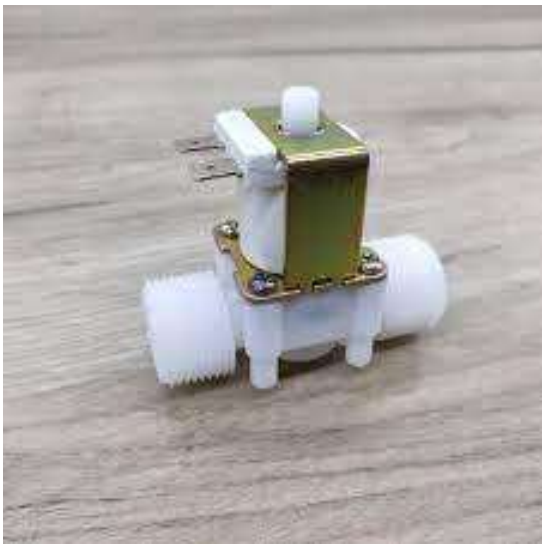


Рисунок 1.5 – Клапан перекриваючого надходження води

В якості клапану перекриваючого надходження води на розпилювач існує велика кількість варіантів. З огляду на відсутність необхідності великої пропускної спроможності клапана, підійдуть навіть найдешевші варіанти клапанів. Електромагнітний клапан складається з еластичної мембрани, розташованої в центрі, яка має жорстке металеве кільце та через пружину з'єднана з плунжером. При подачі напруги на керуючі контакти, під впливом магнітного поля котушки, плунжер піднімається вгору і знімає зусилля з мембрани, яка моментально піднімається та відкриває клапан, відкриваючи при цьому трубопровід для подальшого перебігу рідини. При закритті (відсутності

магнітного) поля), пружний плунжер опускається і з зусиллям притискає мембрану, через кільце до поверхні ущільнювача. Оскільки насос подачі рідини на розпилювач буде вмикатися синхронно з включенням клапана, клапану не доведеться утримувати надмірне тиск, у зв'язку з цим як клапан був обраний нормально закритий клапан китайської розробки, що не має назви.

Характеристики клапана:

- Напруга живлення – 12 вольт
- Утримуваний тиск – до 0,8mpa
- Пропускна здатність – 20 л\хв при тиску 0.8 mpa.
- Споживання струму – 0,03А

Розпилювач типу “Крапельниця” має невеликі габарити, та складається з мінімальної кількості деталей. Завданням розпилювача є розподіл рідини, що надійшла у вхід на маленькі отвори розташування на корпусі розпилювача.

Завдяки такій конструкції збільшується площа поливу. Зі збільшенням тиску в системі довжина струмені рідини з розпилювача буде збільшуватися, охоплюючи велику площу.

1.5 Параметри мікроклімату

1.5.1 Температура

Чим вища температура, тим менше розчиненого кисню залишається у розчині. Падіння вмісту кисню не дуже різке. При температурі від 0° до 30°C вода втрачає приблизно половину свого кисню. У необхідному для росту рослин діапазоні в чистій воді при 20°C є приблизно 9,5 мг/л розчиненого кисню, а при

30°С вміст падає до 7,6 мг/л. Розчинені у воді солі також дещо знижують теоретичний рівень.

В той же час потепління приводить до прискорення метаболізму рослин, що в свою чергу підвищує потребу рослин в кисні, особливо в кореневій зоні, де його поглинання проходить швидше. При температурі до 30°, приблизно, підвищення потреби стає різким.

1.5.2 pH-фактор

Всі рослини вирощуються в дещо кислотній pH, незалежно від того, якому pH вони надають перевагу в ґрунті. Головною причиною цього являється те, що при pH 7 і вище із розчину випадає в осад залізо. Максимально допустимий рівень pH – близько 6,8 з запасом для надійності. Однак в замкнутих системах рекомендується утримувати pH нижче 6,5, щоб запобігти нестачі марганцю. Рослини здатні досить низьких рівнях pH, але рекомендується підтримувати його не нижче 5,5.

Поглинання таких елементів, як Ca, P, Fe, Zn і Mn сильно залежить від pH. Рівень pH можна дещо змінювати в рекомендованих межах для того, щоб не придати переваги ні одному елементу. Занадто точне дотримання рівня pH приводить до внесення в розчин великої кількості хімікатів для корекції, що може порушити рівновагу. Рівень pH має природну схильність підвищуватись. Це відбувається у зв'язку з поглинанням рослин мінералів, та осадження солей.

1.5.3 Електропровідність

Таблиця1.2 - Необхідний рівень електропровідності для різних стадій росту рослин

0,2-0,4	Для черешків

0,8-1,2	Для молодих укорінених рослин
1,6-1,8	Для вегетативної стадії
1,8-2,2	Для стадії цвітіння і плодоношення
2,4-2,6	На останній стадії

Досить часто провідність води для черешків перевищує допустимі межі. В цьому випадку воду слід змішати з неіонізованою водою.

Рівень насичення розчину солями впливає на можливість рослинами живитися водою. Механізмом, який приводить в дію поглинання води коренями, є евапотранспірація, а осмос регулює силу з якою рослини всмоктують воду. Також рівень провідності необхідно коректувати згідно з температурою. При високих температурах рослинам необхідно поглинати багато води, тож провідність необхідно тримати на низькому рівні або навіть дещо нижче рекомендованого рівня. І навпаки, при низьких температурах можна підняти провідність до верхньої межі шкали; невелика транспірація і зменшення поглинання викликають потребу в більш насиченому розчині для того, щоб рослини мали всі необхідні елементи.

Завдяки маніпуляціям з провідністю можна контролювати морфологію рослини. На ранній стадії вегетації, якщо помістити укорінений черешок в середовище з провідністю вище рекомендованої, рослин буде невеликою з малою між вузловою відстанню. І навпаки, якщо провідність занадто низька, рослина буде струнокою, високою і без жорсткої структури.

1.5.4 Вологість

У вологому середовищі листя рослин виростає більшим, ніж в сухому. Максимального росту більшості рослин можна досягнути при відносній вологості 60-80%. Черешкам необхідно близько 90%, а насіння краще всього проростає при

60%. Під час пізнього етапу цвітіння бажано притримуватись нижньої межі шкали або навіть опускатись до 50%, щоб запобігти утворенню плісняви.

Кількість води, яку здатне вміщувати повітря, змінюється в залежності від температури, що приводить до зміни відносної вологості при зміні температури.

1.5.5 Вентиляція

Для забезпечення рослин повітрям та відводу їх відходів життєдіяльності необхідно встановлювати повітрозбірники, котрі, крім того, зможуть використовуватись для контролю вологості і температури. Важливо мати повітрозбірник, котрий зможе витягувати холодне повітря. Крім того, для забезпечення однорідності повітря у великих системах необхідні також циркуляційні вентилятори. Обдування стебла рослини підвищує міцність, а також видуває повітря з-під крони. Це також запобігає поширенню шкідників і хвороб, сприяє транспірації та забезпечує рослини вуглекислим газом.

Повітря треба відновлювати кожні 5 хвилин. Відповідно, якщо прийняти об’єм гідропонної систем за V, витяжний вентилятор повинен буде виробляти V/5 умовних одиниць в годину. До теоретичної величини бажано додавати деякий процент в залежності від сторонніх факторів, таких як фільтри, труби для відведення повітря назовні та інше.

1.5.6 Контроль вуглекислого газу

В процесі фотосинтезу рослині необхідний двоокис вуглецю. Реакція взаємодії виглядає наступним чином:

$$6CO_2 + 12H_2O + \text{фотони} \rightarrow C_6H_{12}O_6 + 6O_2 + 6H_2O$$

Для вивільнення енергії, накопиченої рослиною в процесі фотосинтезу при утворенні органічних речовин, відбувається протилежний процес – дихання:

$6CO_2 + 12H_2O + \text{фотони} \rightarrow C_6H_{12}O_6 + 6O_2 + 6H_2O$

На протязі дня рослини як поглинають CO_2 за допомогою фотосинтезу, так і виділяють його для дихання. Під час темного циклу кисень не виділяється, тільки вуглекислий газ.

Весь газообмін в рослинах проходить через продихи. При високих температурах продихи закриваються і метаболізм рослини сповільнюється. Засвоювання рослинами CO2 також залежить від освітлення. Варто враховувати також необхідність підтримувати певну сприятливу для конкретної культури вологість.

Без належного провітрювання рівень CO_2 швидко падає нижче оптимального, що негативно вплине на метаболізм рослин. В міських умовах вентиляції може бути достатньо для забезпечення потреб рослин. Підвищити рівень CO_2 можна також за рахунок горіння органічних речовин, або ж процесів життєдіяльності (бродіння дріжджів в цукрових розчинах або дихання риб в аквапоніці). Більш дорогими рішеннями є постійна купівля CO_2 в балонах або використання генераторів, котрі за допомогою датчиків CO_2 контролюватимуть рівень автоматично. Використання CO_2 в балонах ускладнюється тим, що скраплений газ в балоні охолоджують до небезпечних для рослин температур. Крім того, використання балони з газом вибухонебезпечні .

Постійно подавати газ не має сенсу: на самому початку світлового дня у рослини закриті продихи. Відкриватися вони починають в середньому через 2 години після світанку (або включення освітлення). Зазвичай подачу здійснюють протягом години (в середньому). В цей час рослина активно поглинає CO2. Після цього генератор бажано відключити в цілях економії.

Повторне підживлення вуглекислим газом рекомендують проводити за 2 години до завершення світлового дня (або відключення ламп). Якщо все контролюється так, щоб рослина звикло до певного циклу, то в цей період у нього як раз відкриваються продихи. В результаті витрата ресурсу стане виправданою.

Далі рослина почне переробляти і засвоювати CO2 протягом ночі. Такий підхід дає максимальний результат при досить економному використанні вуглекислого газу. При проведенні розрахунків бажано мати на увазі, що у різних рослин будуть різні потреби в вуглекислому газі. Для економії вуглекислого газу бажано організовувати окремі гроубокси. Це допомогло б вирішити проблему нераціонального використання ресурсу. В іншому випадку його потрібно буде розпорошувати, орієнтуючись на потреби того рослини, з яким такого газу необхідно найбільше.

Як правило, для трав'янистих рослин збільшення приросту біомаси знаходиться в діапазоні 25-60% у трав'янистих рослин і дещо менше (10-55%) у більшості культурних рослин. Подачу вуглекислого газу важливо збалансувати із подачею інших елементів (вітамінів, мікромінералів, органічних кислот і іншого), інакше буде збільшуватись загальна кількість плодів, їх маса, але зменшуватиметься їх поживна якість.

1.5.7 Освітлення

Сонячне світло - один з найбільш важливих для життя рослин екологічних показників. Завдяки фотосинтезу утворюється приблизно 95% маси сухих речовин рослини. Він поглинається хлорофілом і використовується при побудові первинного органічної речовини. Керування фотосинтезом – один з найефективніших шляхів управління продуктивністю рослин.

Основними характеристиками світла є його спектральний склад, інтенсивність, добова і сезонна динаміка.

Сонячне світло неоднорідне за спектральним складом. З усього спектра для життя рослин важлива фотосинтетична активна (380-710 нм) і фізіологічно активна радіація (300-800 нм).

Найбільше значення мають червоні (720-600 нм) і помаранчеві промені (620-595 нм). Саме вони є основними постачальниками енергії для фотосинтезу

і впливають на процеси, пов'язані зі зміною швидкості розвитку рослини (надлишок червоної і помаранчевої складової спектра затримує перехід рослини до цвітіння).

Сині і фіолетові (490-380нм) приймають безпосередню участь у фотосинтезі, а також стимулюють утворення білків і регулюють швидкість розвитку рослини. У рослин, що живуть в природі в умовах короткого дня, ці промені прискорюють настання періоду цвітіння.

Ультрафіолетові промені з довжиною хвилі 315-380 нм затримують «витягування» рослин і стимулюють синтез деяких вітамінів, а ультрафіолетові промені з довжиною хвилі 280-315 нм підвищують холодостійкість. Лише жовті (595-565 нм) і зелені (565-490 нм) не грають особливої ролі в житті рослин.

Врахування потреб рослин в певному спектральному складі світла необхідний при правильному підборі джерел штучного освітлення

Майже всі декоративні та культурні рослини світлолюбні, тобто краще розвиваються при повному освітлення, але розрізняються по тіньовитривалості. Беручи до уваги ставлення рослин до світла, їх прийнято поділяти на три основні групи: світлолюбні, тіньовитривалі і.

Як і всі живі організми, рослини мають здатність адаптуватися до умов, що змінюються. Ця здатність різна у різних видів. Є рослини, які досить легко пристосовуються до достатнього або надлишкового світла, але зустрічаються і такі, які добре розвиваються тільки при строго визначених параметрах освітленості. В результаті адаптації рослини до зниженої освітленості дещо змінюється його вигляд. Листя стають темно-зеленими і трохи збільшуються в розмірах (лінійні листя подовжуються і стають вже), починається витягування міжвузлів стебла, який при цьому втрачає свою міцність. Потім їх зростання поступово зменшується, тому що різко знижується виробництво продуктів фотосинтезу, що йдуть на ПОБУДОВУ тіла рослини. При нестачі світла багато рослин перестають цвісти.

При надлишку світла хлорофіл частково руйнується, і колір листя стає жовто-зеленим. На сильному світлі зростання рослин сповільнюється, вони виходять більш приосадкуватими з короткими міжвузлями і широкими короткими листям.

Поява бронзово-жовтого забарвлення листя вказує на значний надлишок світла, який шкідливий рослинам. Якщо терміново не вжити відповідних заходів, може виникнути опік.

Важливими характеристиками світлового режиму є добова і сезонна динаміка.

Довжина світлового дня змінюється протягом року. У помірних широтах найкоротший день дорівнює 8 ч., А найдовший - більше 16 год.

Метою дипломної роботи є дослідження та розробка автономної мінітеплиці з дистанційним контролем параметрів. Для досягнення мети треба вирішити наступні задачі:

- дослідити та проаналізувати існуючі аналоги систем автоматичного вирощування рослин; визначити основні параметри мікроклімату, що треба контролювати
- розробити загальну структурну схему управління системою вирощування рослин;
- обрати мікроконтролер, що виконуватиме функції контролю та управління системою згідно з поставленими задачами;
- визначити необхідні датчики для контролю параметрів системи, провести дослідження за результатами якого обрати найбільш відповідні вимогам
- розробити електричну принципову схему та перевірити її працездатність в електронному середовищі Proteus
- розробити прототип приладу

2 РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОГО ВИРОЩУВАННЯ РОСЛИН

2.1 Узагальнена схема системи автоматичного вирощування рослин

Система автоматичного зрощування рослин (Grow box) відстежує різні фактори, що складаються з показників функціонування та управління пристроєм для росту рослин.

Система здатна контролювати:

- тепло – передбачення перегріву або замерзання та їх запобігання;
- воду – контроль рівня та регулярне надходження води за певним графіком;
- світло – освітлення відповідно до графіку;
- вентиляція;
- постачання поживних речовин – за аналізом ґрунту відбувається розподіл поживних речовини за системою вирощування;
- боротьба зі шкідниками шляхом обприскування.

Для автоматизації та ефективного регулювання процесу вирощування рослин, контроль усіх параметрів здійснюється одночасно і становить одну велику систему, яка може бути оптимізована під індивідуальні потреби рослин.

Підбір оптимальних, близьких до ідеальних умов вирощування рослин є важливою частиною, адже на них ґрунтується мікроклімат. Основні завдання системи автоматичного регулювання полягають в:

- керуванні температурою повітря;
- управлінні системою поливу;
- управлінні освітлювальними установками.

Автоматизація процесу вирощування рослин призводить до покращення результатів роботи та не сприяє значному збільшенню собівартості системи .

Узагальнену структурну схему системи автоматичного вирощування рослин можна представити у вигляді сукупності наступних функціональних блоків (рис. 2.1)

Рисунок 2.1 – Узагальнена схема системи автоматичного вирощування рослин

Складові системи автоматичного вирощування рослин (Grow box) включає:

1) Блок імпульсного живлення перетворює постійну напругу в імпульси високої та встановленої шпаруватості, які, як правило, подаються на імпульсний трансформатор. Імпульсні трансформатори виготовляються за таким же принципом, як і низькочастотні трансформатори, тільки в якості осердя використовується не сталь (сталеві пластини), а феромагнітні матеріали - феритові сердечники. не сталь (сталеві пластини), а феромагнітні матеріали - феритові сердечники. Вихідна напруга імпульсного джерела живлення стабілізована, це здійснюється за допомогою негативного зворотного зв'язку, що дозволяє утримувати вихідну напругу на одному рівні навіть при зміні вхідної напруги та навантажувальної потужності на виході блоку. Зворотний негативний зв'язок може бути реалізований за допомогою однієї з додаткових обмоток в імпульсному трансформаторі або за допомогою оптрона, який підключається до

вихідних ланцюгів джерела живлення. Використання оптрона або однієї з обмоток трансформатора дозволяє реалізувати гальванічну розв'язку від мережі змінної напруги. Основні плюси імпульсних джерел живлення (ІДЖ): мала вага конструкції; невеликі розміри; велика потужність; високий ККД; низька собівартість; висока стабільність роботи; широкий діапазон напруги живлення; безліч готових компонентних рішень. До недоліків ІДЖ можна віднести те, що такі блоки живлення є джерелами перешкод, це пов'язано з принципом роботи схеми перетворювача. Для часткового усунення цього недоліку використовують екранування схеми. Також через нестачу деяких пристроїв застосування даного типу джерел живлення є неможливим. Імпульсні джерела живлення стали практично неодмінним атрибутом будь-якої сучасної побутової техніки, що споживає від мережі потужність понад 100 Вт. До цієї категорії потрапляють комп'ютери, телевізори, монітори. Для створення імпульсних джерел живлення, приклади конкретного втілення яких буде наведено нижче, застосовуються спеціальні схемні рішення. Так, для виключення наскрізних струмів через вихідні транзистори деяких імпульсних джерел живлення використовують спеціальну форму імпульсів, а саме біполярні імпульси прямокутної форми, що мають між собою проміжок часу. Тривалість цього проміжку повинна бути більшою за час розсмоктування неосновних носіїв у базі вихідних транзисторів, інакше ці транзистори будуть пошкоджені. Ширина керуючих імпульсів з метою стабілізації вихідної напруги може змінюватись за допомогою зворотного зв'язку. Зазвичай для забезпечення надійності в імпульсних джерелах живлення використовують вьюоковольтні транзистори, які в силу технологічних особливостей не відрізняються на краще (мають низькі частоти перемикання, малі коефіцієнти передачі по струму, значні струми витоку, великі падіння напруги на колекторному переході у відкритому стані).

2) Мікроконтролер (МК) – спеціалізований мікропроцесорний пристрій, орієнтований на виконання керуючих функцій. Інтегруючи на одному кристалі високопродуктивний процесор, пам'ять і набір периферійних пристроїв, мікроконтролери дозволяють з мінімальними витратами реалізувати високоефективні системи і пристрої управління.

3) Датчики:вологості, температури

4) Блок індикації та управління

5) Блок коммутації(Реле)

6) Блок регулювання часу

7) Виконавчі механізми / засоби підтримання мікроклімату: насос, обігрівач, фітолама

2.2 Схема управління системою вирощування рослин

При проектуванні пристроїв доводиться дотримуватися компромісу між розмірами та вартістю схеми управління з одного боку та гнучкістю та продуктивністю з іншого. Для різних випадків оптимальне співвідношення цих та інших параметрів може відрізнятися дуже сильно. Тому існує безліч типів мікроконтролерів, що відрізняються архітектурою процесорного модуля, розміром і типом вбудованої пам'яті, набором периферійних пристроїв, типом корпусу. В даний час однокристальні мікроконтролери є найбільш масовими представниками мікропроцесорної техніки, обсяг випуску яких становить близько 2,5 млрд. штук на рік. Розрізняють наступні типи мікроконтролерів:

- універсальні 16-розрядні МК призначені для реалізації систем реального часу середньої продуктивності. Структура і система команд адаптовані на швидку реакцію за зовнішніми подіями. Найбільше використання мають в системах управління електродвигунами.

- периферійні (інтерфейсні) МК призначені для реалізації найпростіших МП систем управління. Вони мають малу продуктивність і малі габаритні розміри. Зокрема можуть використовуватися периферійними пристроями ЕОМ (клавіатура, миша і т.п.).

- універсальні 8-розрядні МК призначені для реалізації МП систем малої і середньої продуктивності. Мають просту систему команд і велику номенклатуру вбудованих пристроїв.

- цифрові сигнальні процесори (DSP – Digital Signal Processor) призначені для складної математичної обробки вимірюваних сигналів у режимі реального часу. Широко використовуються в телефонії та зв'язку.Основні відмінності DSP: підвищена розрядність оброблюваних слів (16,32,64 біта) і висока швидкість у форматі з плаваючою точкою (16 flops).

- спеціалізовані 32-розрядні МК реалізують високопродуктивну архітектуру і призначені для систем телефонії, передачі інформації, телебачення і інших, що вимагають високошвидкісної обробки інформації.

У сучасних МК використовуються такі типи системи команд процесорів:

- RISC – (Reduce Instruction Set Commands) архітектура зі скороченим набором команд.
- CISC – (Complex Instruction Set Commands) традиційна архітектура з розширеним набором команд.
- ARM – (Advanced RISC – machine) вдосконалена RISC архітектура.

До особливостяей мікроконтролера ATmega328P можна віднести:

-високопродуктивний 8-розрядний мікроконтролер AVR з низьким енергоспоживанням
- розширена архітектура RISC
- 131 потужна інструкція – більшість виконання за один такт
- 32 × 8 робочих регістрів загального призначення
- повністю статична робота
- пропускна здатність до 16 MIPS на частоті 16 МГц
- вбудований двотактний множник
- сегменти енергонезалежної пам'яті високої витривалості
- 32 Кбайт внутрішньосистемної самопрограмованої флеш-пам'яті програм
- 1 Кбайт EEPROM
- 2 Кбайт внутрішньої SRAM
- цикли запису/стирання: 10 000 циклів /100 000 EEPROM
- додатковий розділ коду завантаження з незалежними бітами блокування
- внутрішньосхемне програмування за допомогою програми завантаження на мікросхемі

- справжня операція читання під час запису
- блокування програмування для захисту програмного забезпечення
- периферійні функції
-два 8-бітних таймера/лічильника з окремим режимом попереднього масштабування та порівняння
- один 16-бітний таймер/лічильник з окремим попереднім масштабувальником, режимом порівняння та режим захоплення
- лічильник реального часу з окремим генератором
- шість каналів ШІМ
- 8-канальний 10-бітний АЦП у корпусі TQFP та QFN/MLF
- вимірювання температури
- програмований послідовний USART
- послідовний інтерфейс SPI головний/підлеглий
- байт-орієнтований 2-провідний послідовний інтерфейс (сумісність із Phillips I 2 C)
- програмований сторожовий таймер з окремим вбудованим генератором
- вбудований аналоговий компаратор
- переривання та пробудження після зміни PIN-коду
- спеціальні функції мікроконтролера
- скидання під час увімкнення живлення та програмоване виявлення перегорання
- внутрішній калібрований генератор
- зовнішні та внутрішні джерела переривань
- шість режимів сну: режим очікування, зменшення шуму АЦП, енергозбереження, вимкнення живлення, режим очікування і розширений режим очікування

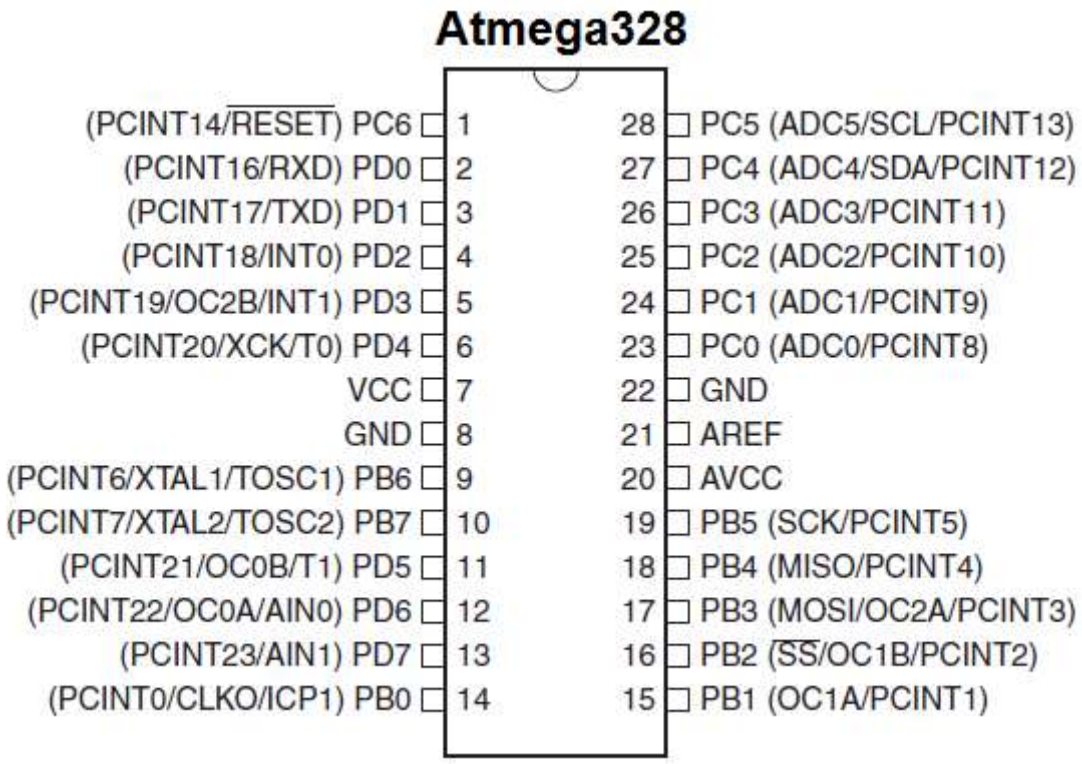


Рисунок 2.2 - Опис та призначення роз'ємів Atmega328

VCC - цифрова напруга живлення.

GND - земля.

Порт B — це 8-розрядний двонаправлений порт вводу/виводу з внутрішніми підтягуючими резисторами (вибираються для кожного біта). Вихідні буфери порту B мають симетричні характеристики приводу з високою здатністю як поглинача, так і джерела. Як входи, штифти порту B, які витягуються ззовні низький буде джерелом струму, якщо підтягуючі резистори активовані. Виводи порту B мають три стани, коли виникає умова скидання активний, навіть якщо годинник не працює.

Залежно від налаштувань запобіжника вибору тактової частоти, PB6 можна використовувати як вхід для підсилювача інвертованого генератора та як вхід для робоча схема внутрішнього годинника.

Залежно від налаштувань запобіжника вибору тактової частоти, PB7 може використовуватися як вихідний сигнал від підсилювача генератора.

Порт C — це 7-розрядний двонаправлений порт вводу/виводу з внутрішніми підтягуючими резисторами (вибираються для кожного біта). Вихідні буфери PC5..0 мають симетричні характеристики приводу з високою здатністю як поглинача, так і джерела. Як входи, контакти порту C, які

витягуються зовні низький буде джерелом струму, якщо підтягуючі резистори активовані. Виводи порту C перебувають у трьох станах, коли виникає умова скидання «активний», навіть якщо годинник не працює.

Якщо запобіжник RSTDISBL запрограмований, PC6 використовується як вхідний контакт. Якщо запобіжник RSTDISBL не запрограмований, PC6 використовується як вхід скидання.

Порт D — це 8-розрядний двонаправлений порт вводу/виводу з внутрішніми підтягуючими резисторами (вибираються для кожного біта). Вихідні буфери порту D мають симетричні характеристики порту з високою здатністю поглинання, так і джерела. Як входи, контакти порту D, які витягуються зовні низький буде джерелом струму, якщо підтягуючі резистори активовані. Виводи порту D мають три стани, коли виникає умова скидання активний, навіть якщо годинник не працює.

AVCC — це контакт напруги живлення для аналого-цифрового перетворювача. Його слід підключити зовні до VCC, навіть якщо АЦП не використовується. Якщо використовується АЦП, його слід підключити до V CC через фільтр нижніх частот. PC6..4 використовує цифрову напругу живлення, VCC.

AREF — аналоговий еталонний контакт для АЦП.

2.3 Блок-схема структури мікроконтролера

Ядро AVR поєднує багатий набір інструкцій із 32 робочими регістрами загального призначення. Усі 32 регістри безпосередньо підключені до арифметико-логічного пристрою (ALU), дозволяючи отримати доступ до двох незалежних регістрів в одній інструкції за один такт. Отримана архітектура є більш ефективною з використанням коду та забезпечує пропускну здатність у десять разів швидше, ніж звичайні мікроконтролери CISC.

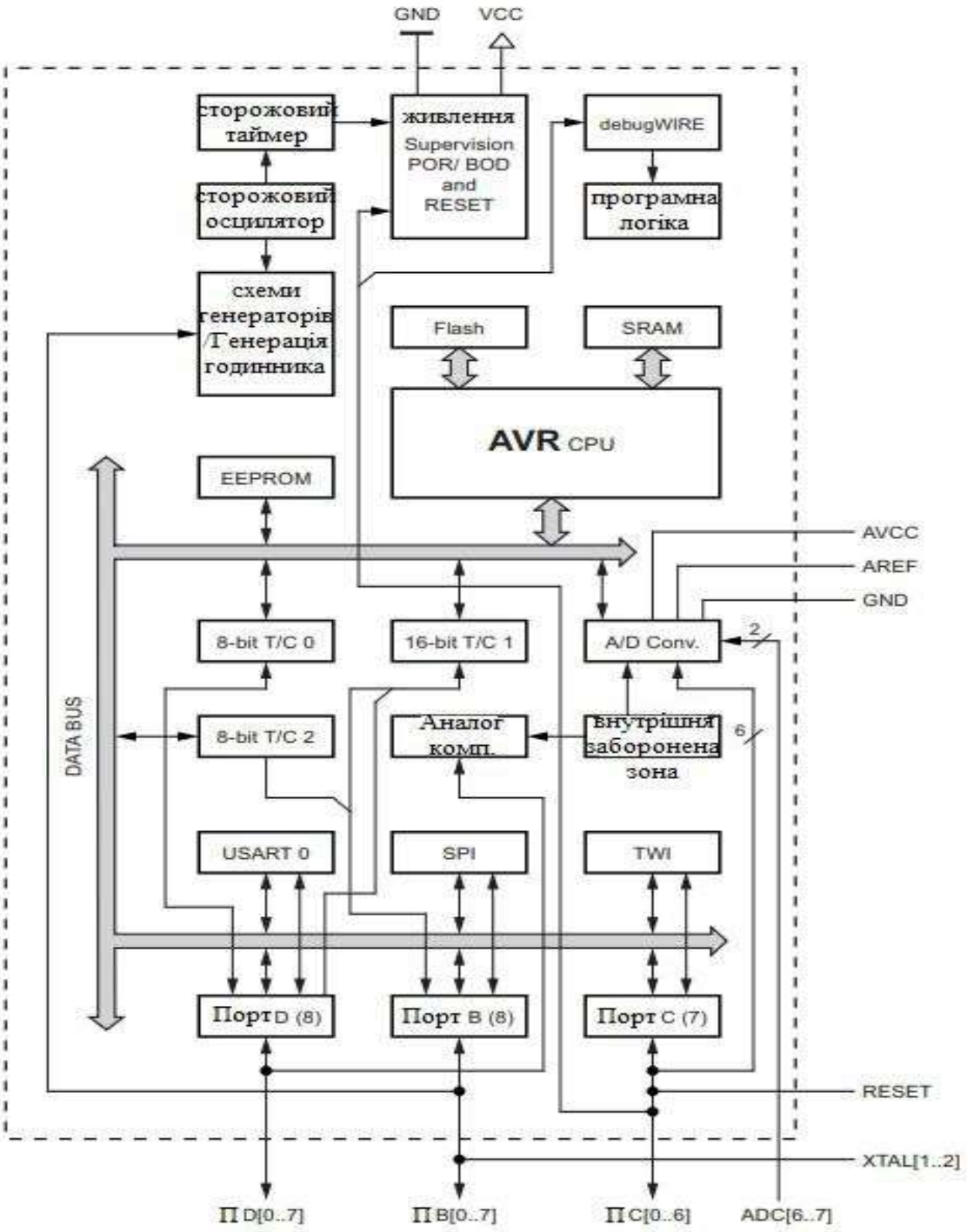


Рисунок 2.3 – Структура мікроконтролера

Atmel ATmega328P забезпечує наступні функції: 32 Кбайт внутрішньосистемної програмованої флеш-пам'яті з можливотю читанням під час запису, 1 Кбайт EEPROM, 2 Кбайт SRAM, 23 лінії вводу/виводу загального призначення, 32 робочі регістри загального призначення, три гнучких таймера/лічильника з режимами порівняння, внутрішні та зовнішні переривання, послідовний програмований USART, байт-орієнтований 2-провідний послідовний інтерфейс, послідовний порт SPI, 6-канальний 10-бітний АЦП (8

каналів у корпусах TQFP і QFN/MLF), програмований сторожовий таймер із внутрішнім генератором і п'ять режимів енергозбереження, які можна вибрати програмно. Режим очікування зупиняє ЦП, дозволяючи SRAM, таймеру/лічильникам, USART, 2-провідному послідовному інтерфейсу, порту SPI та системі переривань працювати та продовжувати функціонування. Режим ввімкнення/вимкнення зберігає вміст регістру, але сповільнює(“фрізить”) генератор, відключаючи всі інші функції мікросхеми до наступного переривання або апаратного ресету(reset) У режимі енергозбереження асинхронний таймер продовжує працювати, дозволяючи користувачу підтримувати базу таймера, поки сам пристрій перебуває в режимі сну. Режим шумозаглушення АЦП зупиняє ЦП і усі модулі вводу/виводу, крім асинхронного таймера та АЦП, щоб мінімізувати шум перемикання під час перетворення АЦП. У режимі очікування кварцевий/резонаторний генератор працює, а решта пристрою перебуває в режимі сну. Це забезпечує дуже швидкий запуск в поєднанні з низьким енергоспоживанням.

Пристрій виготовлено за технологією енергонезалежної пам'яті високої щільності Atmel. Флеш-пам'ять ISP на чіпі дозволяє перепрограмувати пам'ять програми в системі через послідовний інтерфейс SPI за допомогою звичайної енергонезалежної пам'яті програматора або за допомогою вбудованої програми завантаження, що працює на ядрі AVR. Програма завантаження(boot program) може використовувати будь-який інтерфейс для завантаження у флеш-пам’ять програми. Програмне забезпечення в розділі boot-флеш-пам’яті продовжуватиме працювати, поки у цьому розділі оновлено програми, забезпечуючи реальну операцію “read-while-write” під час запису. Завдяки поєднанню 8-розрядного процесора RISC з внутрішньосистемною самопрограмованою флеш-пам’яттю на монолітному чіпі.

Atmel ATmega328P – це потужний мікроконтролер, який забезпечує дуже варіативність та економічно ефективне рішення для багатьох програм вбудованого керування. ATmega328P AVR підтримується повним набором інструментів розробки програм і систем, включаючи: компілятори C, асемблери

макросів, налагоджувачі/симулятори програм, внутрішньосхемні емулятори та комплекти оцінки

2.4 Архітектура ядра AVR

Основною функцією ядра процесора є забезпечення правильного виконання програми. Тому центральний процесор повинен мати доступ до пам'яті, виконувати обчислення, керувати периферійними пристроями тощо обробляти переривання.

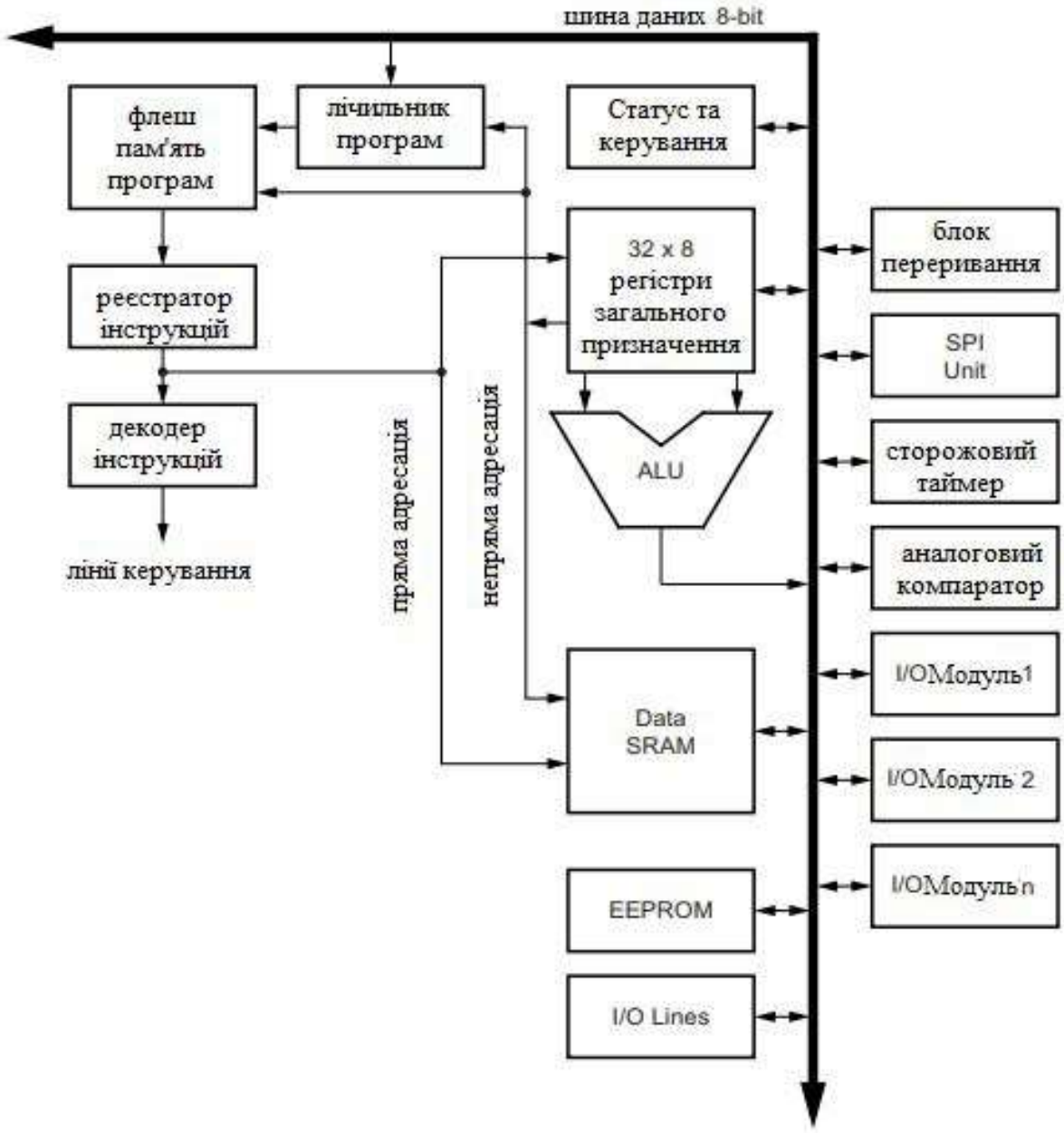


Рисунок 2.4 - Блок-схема архітектури AVR

Щоб максимізувати продуктивність і паралельне виконання, AVR використовує гарвардську архітектуру – з окремою пам’яттю та шини для програм і даних. Інструкції в програмній пам'яті виконуються за допомогою однорівневої конвеєрної обробки. Поки одна інструкція виконується, наступна інструкція попередньо вибирається з пам’яті програми. Ця концепція дозволяє інструкції виконуватись у кожному такті. Програмна пам'ять — флеш-пам'ять, що перепрограмується в системі. Регістровий файл швидкого доступу містить 32 × 8-розрядні робочі регістри загального призначення з одним часом доступу тактового циклу. Це дозволяє працювати з арифметико-логічним пристроєм (АЛП) за один цикл. У типовій операції ALU два операнди виводяться з реєстрового файлу, операція виконується, а результат зберігається– за один такт.

Шість із 32 регістрів можна використовувати як три 16-розрядні вказівники регістрів непрямих адресів для адресації набору даних, що дозволяє ефективні обчислення. Один із цих адресних показників також можна використовувати як адресний показник для пошуку таблиць програм флеш пам'яті. Ці додані функціональні регістри є 16-розрядними X-, Y- та Z-регістрами. ALU підтримує арифметичні та логічні операції між регістрами або між константою та регістром. Єдиний реєстр операції також можна виконувати в ALU. Після виконання арифметичної операції регістр стану оновлюється для відображення інформація про результат операції.

Потік програми забезпечується умовними та безумовними інструкціями переходу та виклику, здатними безпосередньо звертатися до цілого адресного простору. Більшість інструкцій AVR мають один 16-бітний формат слова. Кожна адреса пам'яті програми містить 16- або 32-розрядну інструкцію.

Простір програмної флеш-пам’яті розділено на дві частини: програму завантаження та програму прикладної програми. Обидва розділи мають спеціальні біти блокування для захисту від запису та читання/запису. Інструкція SPM, яка записується в програму флеш пам'яті повинен знаходитися в розділі «програми завантаження». Під час викликів переривань і підпрограм програмний лічильник адреси повернення (PC) зберігається в стеку. Стек ефективно

розподіляється в SRAM загальних даних, і, отже, розмір стека обмежений лише загальним розміром SRAM і використання SRAM. Усі програми повинні ініціалізувати SP у процедурі скидання. Показник стека (SP) доступний для читання/запису в просторі введення/виведення. Через SRAM даних можна легко отримати доступ п'яти різних режимів адресації, які підтримуються в архітектурі AVR. Усі простори пам’яті в архітектурі AVR є лінійними та регулярними картами пам’яті.

Гнучкий модуль переривань має регістри керування в просторі вводу-виводу з додатковим бітом дозволу глобального переривання в статусі зареєструватися. Усі переривання мають окремий вектор переривань у таблиці векторів переривань. Переривання мають пріоритет відповідно з позицією вектору переривання. Чим нижче адреса вектор переривання, тим вищий пріоритет. Простір пам'яті вводу-виводу містить 64 адреси для периферійних функцій центрального процесора, таких як регістри керування, SPI та інші функції вводу-виводу. Доступ до пам’яті вводу/виводу можна отримати безпосередньо або як розташування простору даних, наступне за файлом реєстру.

ATmega328P розроблено та виготовлено відповідно до найсуворіших міжнародних вимог стандарт ISO-TS-16949. Якість і надійність ATmega328P були перевірені під час регулярного продукту кваліфікація згідно з AEC-Q100 клас 1.

2.5 Підключення датчиків температури та вологості

Для вибору датчика темтератури та вологості було досліджено п’ять популярних модулів Aht10, DHT11, DHT22, HDC1080, sht30

Таблиця 2.2- Порівнянна датчиків температури та вологості

Назва датчику	AHT10	HDC1080	DHT11	DHT22	sht30
Діапазон вимірвання температури	-40°C … +85°C ±0.3 °C	от -40 до + 125°C ± 0.2°C	-20 ~ +60 ºC ± 2%	-40 ~ 80 ºC ± 0.5	-40...125 C +/- 0.3 C
Діапазон вимірюання вологи	0 … 100% RH ± 2%	0-100% RH ± 2%	5 - 95% RH ± 5%	0-100% RH ± 2%	0 % - 100 % RH +/- 3 %
Інтерфейс	I2C	I2C	1-wire	1-wire	I2C
Напруга живлення	2,0 — 5,5 В	2.7 - 5.5 В	3.5-5.5 В	3.6-6 В	2,4...5.5 В.
Габаритні розміри	15,5x11 мм	16x15,8мм	15.5 x 12 мм	25.1 x 15.1 мм	16x13 мм
Оріентовна ціна	98,00	146.00 грн	49 грн	152 грн	205,00

DHT11 та DHT22 (AM2302) є цифровими датчиками температури, що вимірюють і температуру і вологість. Вони виглядають дуже схожим і працюють однаково, але мають різні характеристики. Обидва датчики можуть живитися від 3,3 або 5 В. Датчик DHT22 має кращу роздільну здатність і більш широкий діапазон вимірювання температури та вологості. Хоч він має більшу вартість, але інтервал зчитування 2 секунди. DHT11 дешевший, має менший діапазон і менш точний. Він має інтервал зчитування 1 секунду .

AHT10 – це цифровий датчик температури та вологості. Дворядний плоский безвивідний SMD-корпус має розміри 4x5мм та висоту 1,6мм. Датчик AHT10 представляє собою мікросхему ASIC нової конструкції з покращеним напівпровідниковим ємнісним датчиком вологості MEMS та стандартним вбудованим датчиком температури. Він може виводити калібрований цифровий сигнал по I2C шині. Датчик має діапазон живлення від 1,8 до 3,6 В, але робоча напруга, що рекомендується, - 3,3 В. Оскільки цей датчик призначений для штампування на спеціальній друкованій платі за допомогою процесу сплавлення, раціональніше для монтажу обрати модуль. Окрім цього на модулі

встановлено стабілізатор 3,3 В LM6206-3.3/XC-6206-3.3, що являє собою трехвивідний низькоточний стабілізатор, виконаний за CMOS технології. Стабілізатор може забезпечити максимальний вихідний струм до 100 мА і допускає вхідну напругу до 6 В. Також на модулі встановлено зсув логічного рівня який складається з масиву SMD резисторів 10 кОм х 4 (103) та 6-контактного здвоєного N-канального MOSFET-чіпа що дозволяє підключати його до логічного рівня в 5в.

HDC1080 - це цифровий датчик від Texas Instruments Incorporated являє собою датчик температури та вологості

sht30- це цифровий датчик температури та вологості від Sensirion

З порівняння глобальних пріоритетів різних датчиків (розділ 3) видно, що найбільшим є пріоритет у варіанта датчика температури та вологості AHT10.

Ємнісний датчик є друкована лезоподібна плата , яка буде занурюватися в ґрунт до 80 мм . На ній дві доріжки двох електродів, які захищені токоізолюючою маскою і не піддаються корозії (Рис 2.5)

Рисунок 2.5- Ємнісний датчик

На платі ємнісного датчика знаходиться RC-генератор побудований на таймері ME555 частота якого залежить від ємності між двома електродами.

Зміна вологості ґрунту позначається на його діелектричних властивостях та змінює ємність, що призводить до підвищення або зниження вихідного сигналу датчика. Підсумкова напруга пропорційна ступеню вологості ґрунту.

Dallas DS18B20 з повністю аналогічними параметрами: діапазон вимірюваних температур від -55 до +125 °C. Зчитуваний з приладу цифровий код є прямим безпосереднім кодом виміряного значення температури і не потребує додаткових перетворень. Програмована користувачем роздільна здатність вбудованого АЦП може бути змінена в діапазоні від 9 до 12 розрядів вихідного коду. Абсолютна похибка перетворення менше 0.5 °C в діапазоні контрольованих температур -10 до +85 °C. Максимальний час повного 12-ти розрядного перетворення ~750 мс (при роздільній здатності 12 розрядів). Для підключення потрібно резистор 4.7 кОм. Внутрішня енергонезалежна пам'ять температурних установок забезпечує запис довільних значень верхньої та нижньої межі установок. Крім того, мікросхема містить вбудований логічний механізм пріоритетної сигналізації у випадку виходу температури за один з обраних порогів. Вузол 1-Wire-інтерфейсу приладу організований таким чином, що існує теоретична можливість адресації необмеженої кількості подібних пристроїв на однопровідній лінії. Термометр має індивідуальний 64-розрядний реєстраційний номер (груповий код 028H) і забезпечує можливість роботи без зовнішнього джерела живлення, тільки за рахунок паразитного живлення однопровідної лінії. Живлення приладу через окремий зовнішній вивід здійснюється напругою від 3.0В до 5.5В.

2.6 Розробка блоку комутації

Для передачі посилених по потужності сигналів по зонах оповіщення, а також для забезпечення контролю відсутності короткого замикання чи обриву в лініях зон оповіщення розроблено блок комутації, який складається з двох блоків. Перший блок виконує комутування високої напруги та включає два

симістори, оптопару для гальванічної розвязки та Снабер – пристрій для уникнення стрибків напруги в електричних системах.

Оптосимістор належать до класу оптронів і забезпечують дуже хорошу гальванічну розв'язку (близько 7500 В) між керуючим ланцюгом та навантаженням (рис.2.6). Ці радіоелементи складаються з арсенід-гелієвого інфрачервоного світлодіода, з'єднаного за допомогою оптичного каналу з двонаправленим кремнієвим перемикачем.

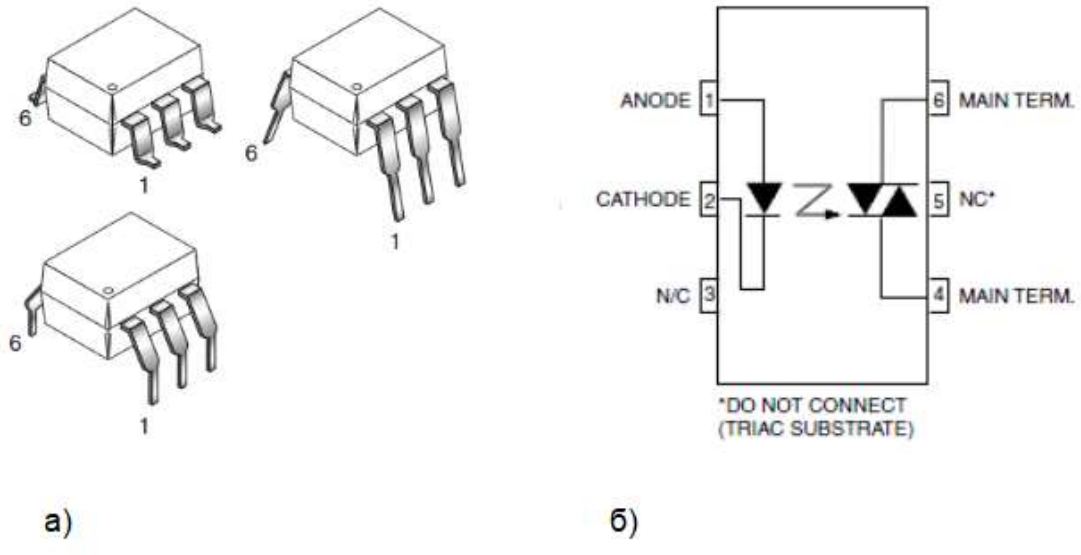


Рисунок 2.6 – Зовнішній вигляд (а) та призначенння виводів (б) оптопари з симисторним входом MOC3023

Таблиця 2.3 – Характеристики оптопари MOC3023

№	Параметр	Значення
1	Напруга ізоляції	7.5 кВ
2	Зворотна напруга світлодіода	3 В
3	Прямое напряжение светодиода	1.2 В
4	Пряма напруга світлодіода	60 мА
5	Вхідний струм спрацьовування	3…5 мА
6	Пікова напруга симистора	400 В
7	Вихідний струм увімкнення симістора при 25 C^o	100 мА
8	Вихідний струм увімкнення симістора при 75 C^o	50 мА

9	Піковий вихідний струм	1.2 А

Блок комутувння високої напруги представлено на рисунку 2.7

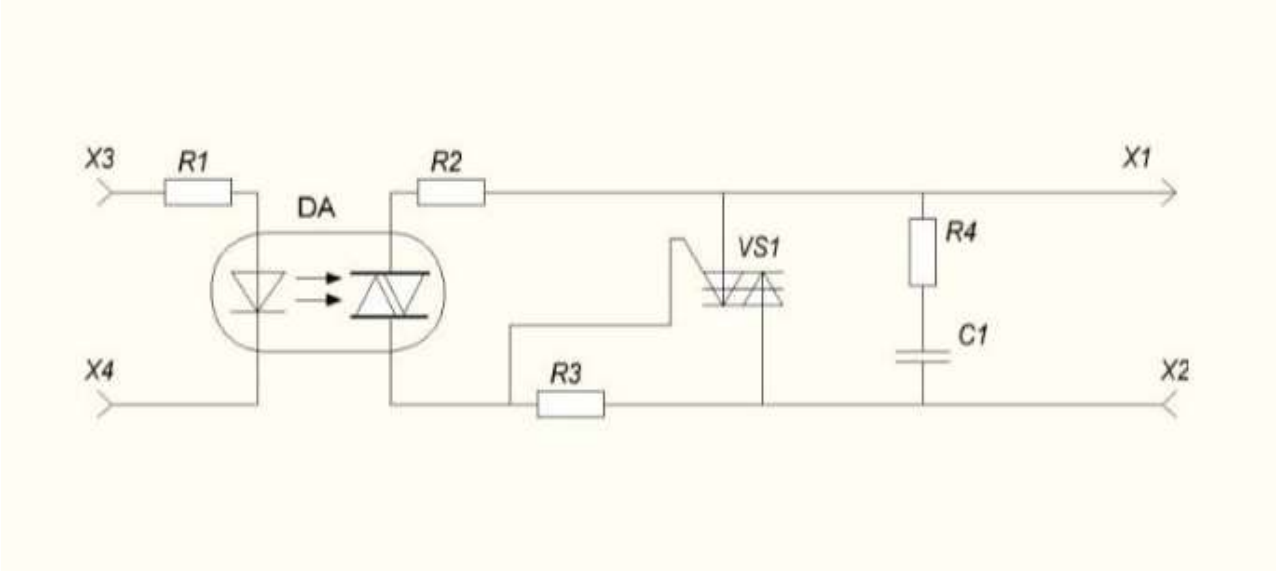


Рисунок 2.7 – Блок комутувння високої напруги

Розрахахуємо величину опору резистора R1. Опір обмежувального резистора R1 залежить від мінімального прямого струму інфракрасного світлодіода, необхідного для спрацювання симістору. Розрахуємо R діода для оптосимістора MOC3023 и напруги живлення +5 вольт. В даному випадку максимальний струм, який зможе пропустити світлодіод оптосимістора дорівнює значенню 60 мА, робочий струм 5 мА. Необхідно врахувати струм світлодіода 10 мА с урахуванням зниження ефективності світлодіоду протягом строку служби, поступового послаблення сили струму (запас 5 мА).

Таким чином R1 = (5-1,5)/0,01 = 350 Ом (обираємо 360 Ом).

При використанні мікроконтролера, необхідно враховувати падіння напруги порядку 0,3 вольта та розрахунки проводити не для 5 вольт, а для 4,7 вольт. В такому випадку R1 складає 320 Ом (обираємо 330 Ом).

Резистор R2 на схемі вмикати необов'язково, якщо навантаження тільки резистивне. Однак, якщо симістор захищений ланцюгом R4-C2, резистор R2 дозволяє обмежити струм через керуючий електрод оптосимістора.

У разі індуктивного навантаження струм і напруга, що проходить через симистор, знаходяться в протифазі. Так як симістор перестає бути провідником, коли струм проходить через нуль, конденсатор С2 може розряджатися через оптосимістор. Тоді резистор R2 обмежить цей струм розряду. Знаючи, що максимально допустимий струм для оптосимістора MOC3023 дорівнює 1 ампер і прийнявши за максимальне значення напруги, що діє, в мережі 260 вольт, розрахуємо мінімальне значення опору R2:

$R2 = 260 \cdot \sqrt{2} / 1 = 368$ Ом (обираємо 360 Ом).

Занадто велика величина може призвести до порушення роботи.

Значення резистора R3 становить 330 Ом. Резистори R2 і R3 вводять затримку відмикання симістора, яка буде тим значнішою, чим вищий опір цих резисторів.

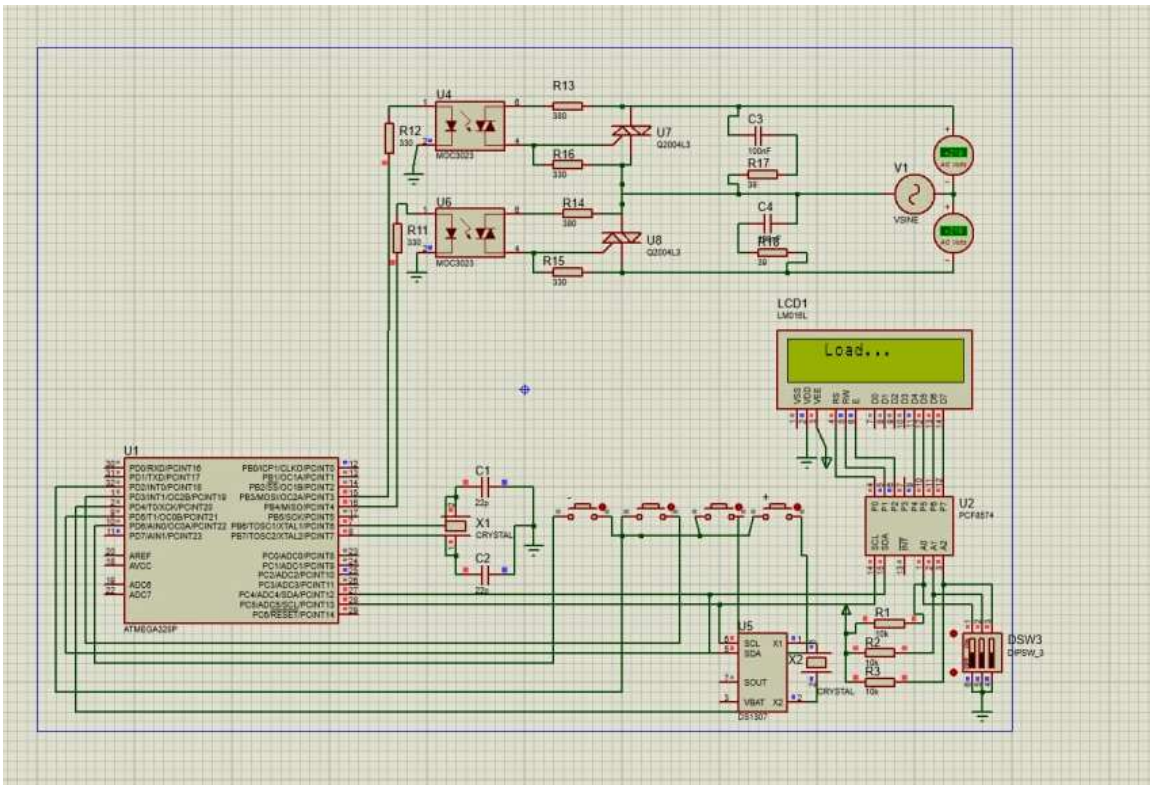


Рисунок 2.8– перевірка роботи блоку комутувння високої напруги

Блок комутації низької напруги складається з струмообмежувального резистора R1(100Ом) що захищає вхід контролера від перевищених значень сили струму, резистор R2 (10КОм) автоматично закриває транзистор за відсутності сигналу з МК і польового транзистора VT1 (70to3gh). Переваги польового

транзистора в порівнянні з біполярним є очевидною. На затвор подається напруга, але при наявності діелектрика струм буде нульовим, а отже необхідна потужність на керування цим транзистором буде ммінімальною, за фактом він споживає тільки в момент перемикання, коли йде заряд і розряд конденсатора.

Недоліком є його ємнісні властивості - наявність ємності на затворі вимагає великого зарядного струму при відкритті. Теоретично це дорівнює нескінченності на нескінченно малому проміжку часу. А якщо струм обмежити резистором, то конденсатор буде заряджатися повільно.

Під час комутації індуктивного навантаження відбувається викид напруги, який може пошкодити транзистор. Для захисту від нього в колі підключено паралельно діод.

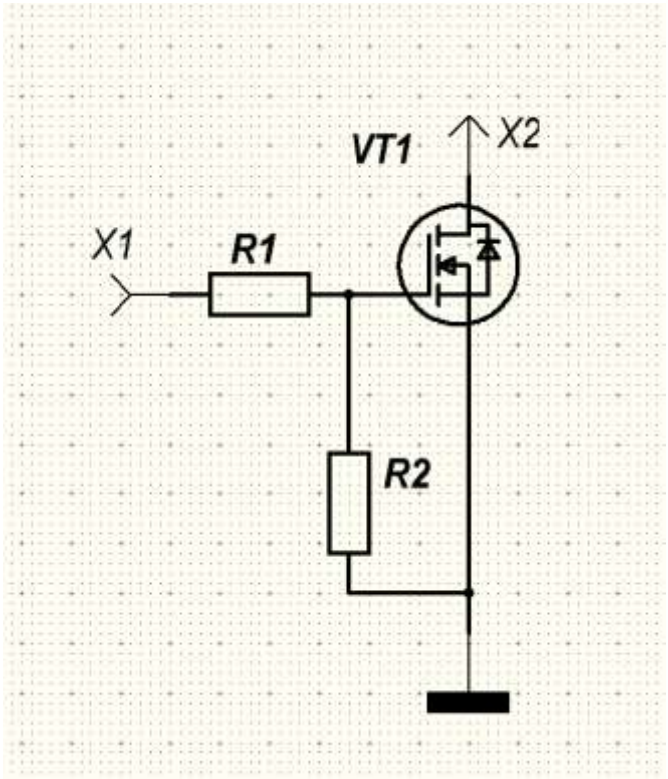


Рисунок 2.9 – Блок комутувння низької напруги

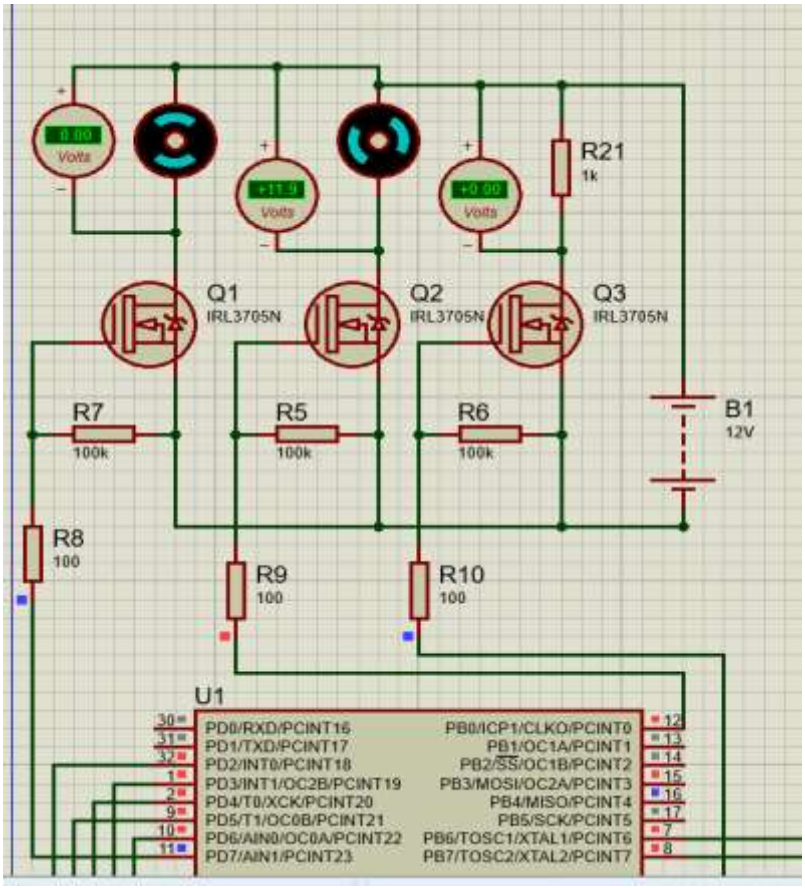


Рисунок 2.10 – Підключення блок комутувння низької напруги до мікроконтроллера

2.7 Використання двунаправленої шини передачи даних

На рисунку 2.11 представлено моделювання схеми в електронному середовищі Proteus з використанням шини 1-Wire. 1-Wire — шина передачі даних пристроїв, розроблена компанією Dallas Semiconductor Corp., що дозволяє здійснювати передачу даних на малій швидкості, сигналізацію, і живлення через єдиний провідник.

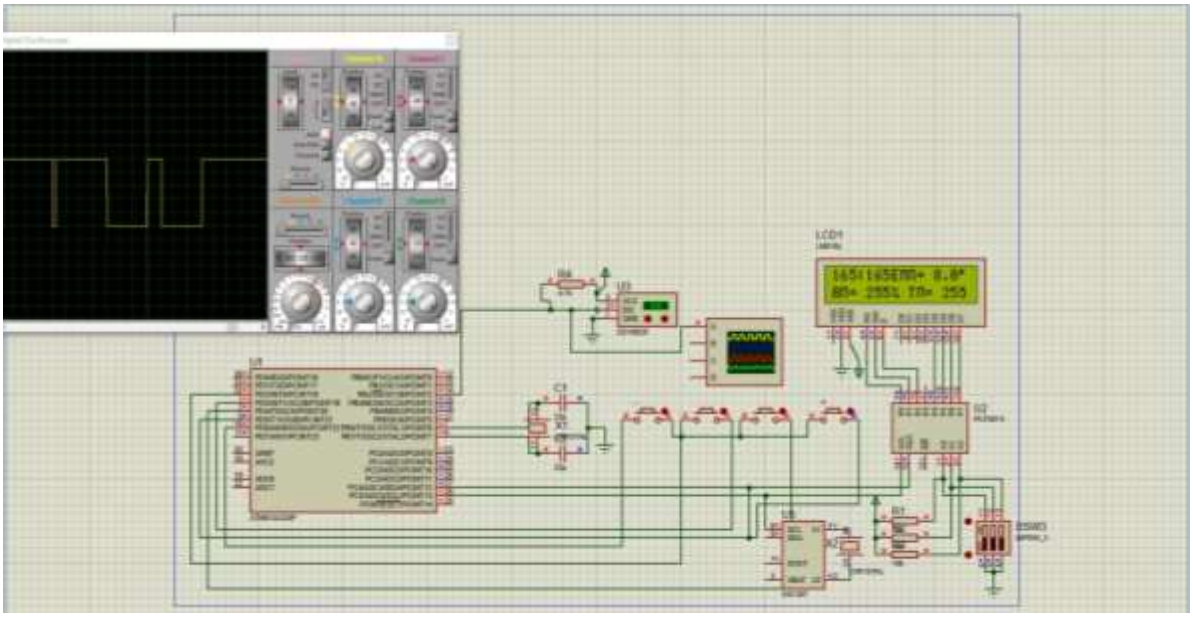


Рисунок 2.11 - Реалізація шини 1-Wire

Шина I2C є синхронною, складається з двох ліній: даних (SDA) та тактування (SCL). Є ведучий (master) та ведені (slave). Ініціатором обміну завжди виступає ведучий, обмін між двома веденими неможливий. Усього на одній двопровідній шині може бути до 127 пристроїв. Такти лінії SCL генерує master. Лінією SDA можуть керувати як майстер, так і ведений залежно від напряму передачі. Одиницею обміну є пакет, обрамлений унікальними умовами на шині, іменованими стартовим і стоповим умовами. Майстер на початку кожного пакета передає один байт, де вказує адресу веденого та напрямок передачі наступних даних. Після кожного слова передається один біт підтвердження прийому приймальною стороною.

I²C використовує дві двонаправлені лінії, підтягнуті до напруги живлення і керовані через відкритий колектор або відкритий стік - послідовна лінія даних (SDA, англ. Serial DAta) та послідовна лінія тактування (SCL, англ. Serial CLock). Стандартні напруги +5 або +3,3 В, однак допускаються й інші. Класична адресація включає 7-бітовий адресний простір із 16 зарезервованими адресами. Це означає, що розробникам доступно до 112 вільних адрес для підключення периферії на одну шину. Основний режим роботи - 100 кбіт/с; 10 кбіт/с у режимі роботи зі зниженою швидкістю. Також важливо, що стандарт допускає припинення тактування для роботи з повільними пристроями.

PCF8574 містить 8-бітний порт введення-виведення загального призначення, читання або запис даних здійснюється будь-яким мікроконтролером або іншим пристроєм по шині I2C. Розширювач має низький споживаний струм і виходи з регістром з високими характеристиками струму для прямої передачі сигналу на світлодіоди і т.п. Також у пристрої є лінія переривання (INT), яка може бути підключена до логіки переривання мікроконтролера. Посилаючи сигнал переривання цією лінією, дистанційне введення - висновок повідомляє мікроконтролеру про що надходять його порти даних, без необхідності підтримувати зв'язок через I2C-шину. Це означає, що PCF8574 може залишатися простим "підпорядкованим" пристроєм.

На рисунку 2.12 представлено моделювання схеми в електронному середовищі Proteus з використанням шини I²C

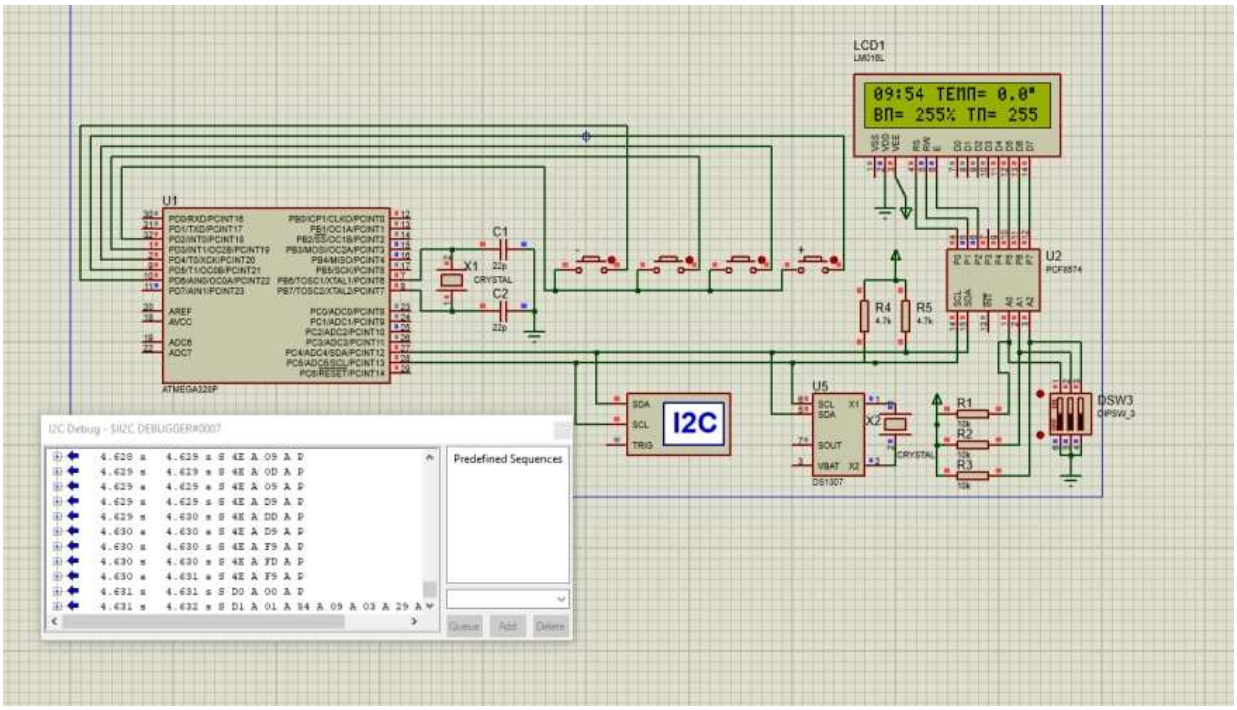


Рисунок 2.12 - Реалізація шини I²C

2.8 Розробка схеми блоку живлення

Мікросхема IR2153 є високовольтним драйвером затвора, на ній будують багато різних схем, блоки живлення, зарядні пристрої і т. д. Напруга живлення може змінюватись від 10 до 20 вольт, робочий струм 5 мА і робоча температура до 125 градусів Цельсія. Це стандартне напівмостове джерело живлення з IR2153. Діодний міст на вході 1n4007 або готова діодна збірка розрахована на струм не менше 1 А із зворотньою напругою 1000 В. Резистор R1 не менше 2 Ватт 24 кОм, резистор R2, R3, R4 потужністю 0,25 Ватт.

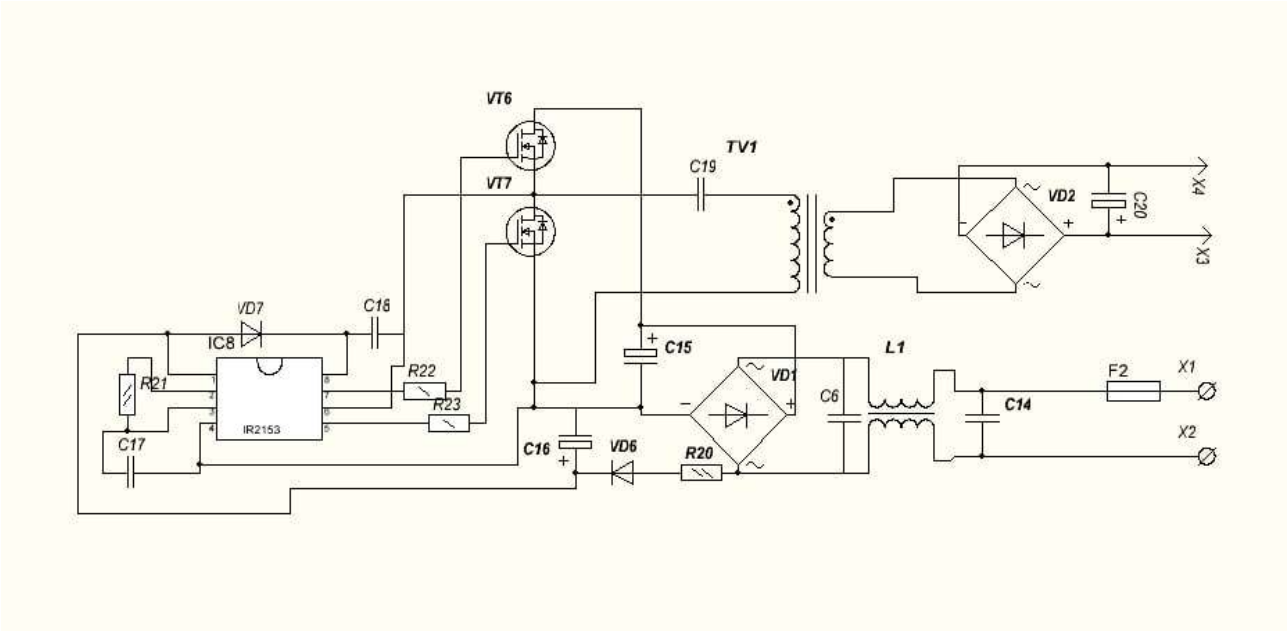


Рисунок 2.13 – Схема блоку живлення

Конденсатор електролітичний з високої сторони 400 вольт 47 мкф. Вихідний 35 вольт 470 - 1000 мкф. Конденсатори фільтра плівкові розраховані на напругу не менше 250 В 0,1 - 0,33 мкФ. Конденсатор С5 – 1 нФ. Керамічний, конденсатор С6 керамічний 220 нФ, С7 плівковий 220 нФ 400 В. Транзистор VT1 VT2 марки N IRF840, діодний міст на виході повноцінний з чотирьох ультрашвидких діодів HER308 або інші аналогічні.

Перевага цієї схеми в тому, що схема має достатньо високий ККД та невеликі розміри. Схема живиться від напруги 220 вольт, має на вході фільтр, який

складається з дроселя і двох плівкових конденсаторів розрахованих на напругу не менше 250 - 300 Вольт ємністю від 0,1 до 0,33 мкФ. Далі напруга надходить на діодний міст, розрахований на зворотну напругу не менше 400 Вольт і струм не менше 1 Ампера. Можна використати готову діодну збірку. Далі за схемою стоїть конденсатор, що згладжує, з робочою напругою 400 В, оскільки амплітудне значення мережевої напруги становить приблизно 300 В. Ємність даного конденсатора підбирається з розрахунку 1 мкФ на 1 Ватт потужності. Живлення мікросхеми є змінним, резистор R1 забезпечує гасіння струму, бажано не менше двох ват тому що здійснюється його нагрівання, потім напруга випрямляється всього одним діодом на конденсатор, що згладжує, потім передається на мікросхему. В схемі мікросхема працює на частоті 47 - 48 кГц. Для такої частоти організована RC ланцюжок що складається з резистора R2 15 ком та плівкового або керамічного конденсатора на 1 нФ.

При такому розкладі деталей мікросхема працюватиме правильно і вироблятиме прямокутні імпульси на виході, які надходять на затвори потужних польових ключів через резистори R3, R4. Номінали їх можуть відхилятися в межах від 10 до 40 Ом. Транзистори необхідно ставити N канальні, в даному випадку використовується IRF840 з робочою напругою стік виток 500 В і максимальним струмом стоку при температурі 25 градусів 8 А і максимальною потужністю, що розсіюється 125 Ватт. Далі за схемою стоїть імпульсний трансформатор, після нього йде повноцінний випрямляч з чотирьох діодів марки HER308, звичайні діоди не зможуть працювати на високих частотах, тому треба використовувати ультрашвидкі діоди і після мосту напруга надходить на вихідний конденсатор 30 Вольт. Можна обрати 470 мкФ. Особливо великих ємностей в імпульсних блоках живлення не потрібно. З розрахунку того, що на виході потрібно отримати напругу приблизно 12-14 Вольт, первинна обмотка трансформатора містить 47 витків проводом 0,6 мм у дві жили з ізоляцією між намотуванням, вторинна обмотка містить 4 витка того ж дроту в 7 жил.

2.9 Розробка друкованої плати

Друкована плата являє собою пластину яка виконана з діелектрика (найбільш часто використовуються такі матеріали, як текстоліт, склотекстоліт, гетинакс), на якій сформовано (зазвичай друкованим методом) електропровідний ланцюг. Друкована плата використовується для електричного і механічного з'єднання різних електронних компонентів або з'єднання окремих електронних вузлів. Електронні компоненти на платі з'єднуються своїми виводами з елементами провідного рисунка, зазвичай паянням в результаті чого збирається електронний модуль. На відміну від навісного монтажу, на друкованій платі електропровідний малюнок виконаний з фольги адитивним або субтрактивним методом. Друкована плата зазвичай містить монтажні отвори і контактні площадки, які можуть бути додатково покриті захисним покриттям: сплавом олова і свинцю, оловом, золотом, сріблом, органічним захисним покриттям. Крім того в друкованих платах є перехідні отвори , зовнішнє ізоляційне покриття яке закриває ізоляційним шаром невживану для контакту поверхню плати, маркування зазвичай наноситься за допомогою шовкографії, рідше - струменевим методом або лазером.

Для цього проекту було Трасовано 2 плати у середовищі Sprint layout від компанії розробника ABACOM-Ingenieurgesellschaft (Німеччина). На першій частині плати було розташовано мікроконтролер, блок годинника реального часу та блок комутації. На другій частині було розташовано елементи блока живлення. Плата конвертора pcf8674 являє собою окремий модуль

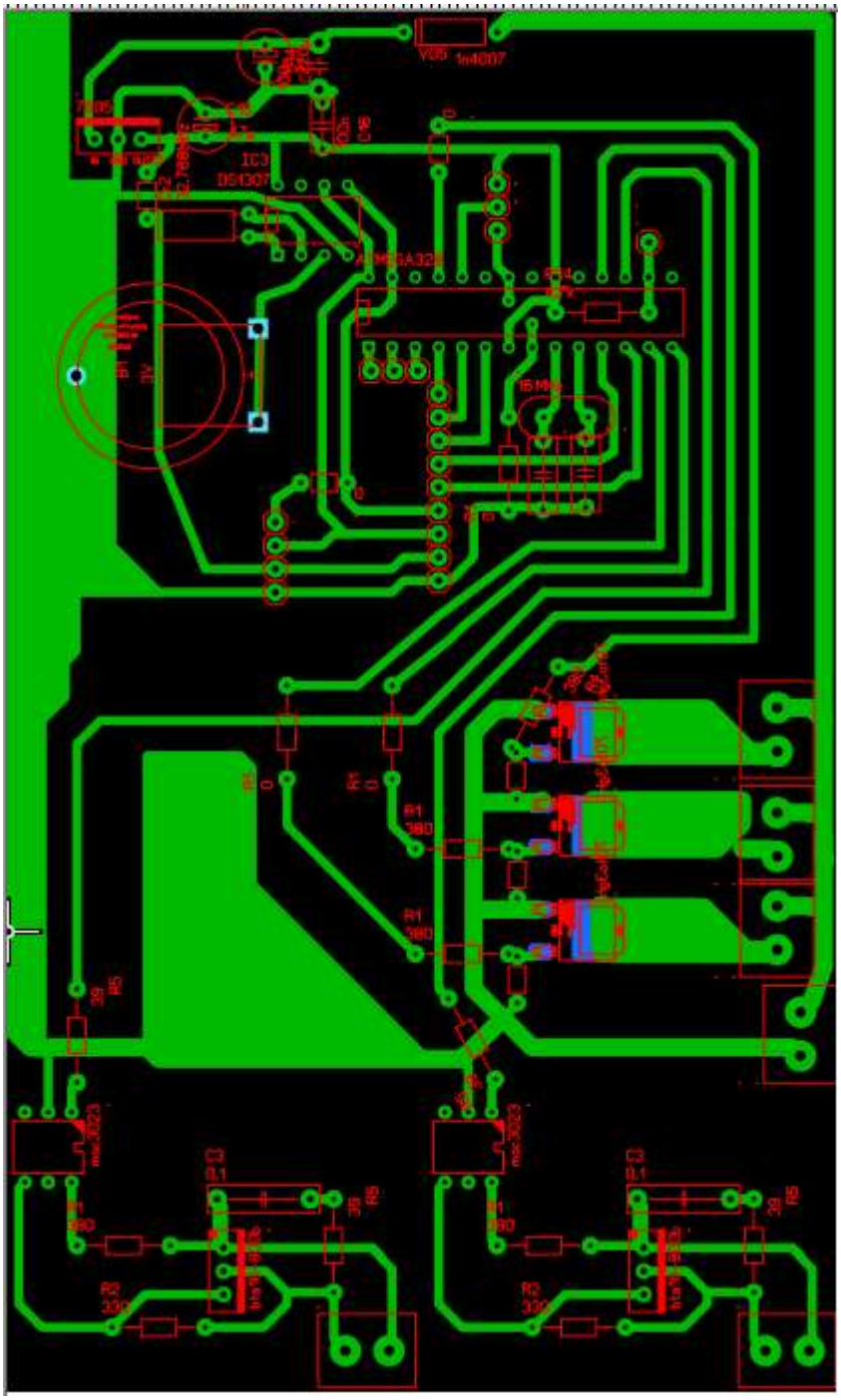


Рисунок 2.14 – Основна друкована плата

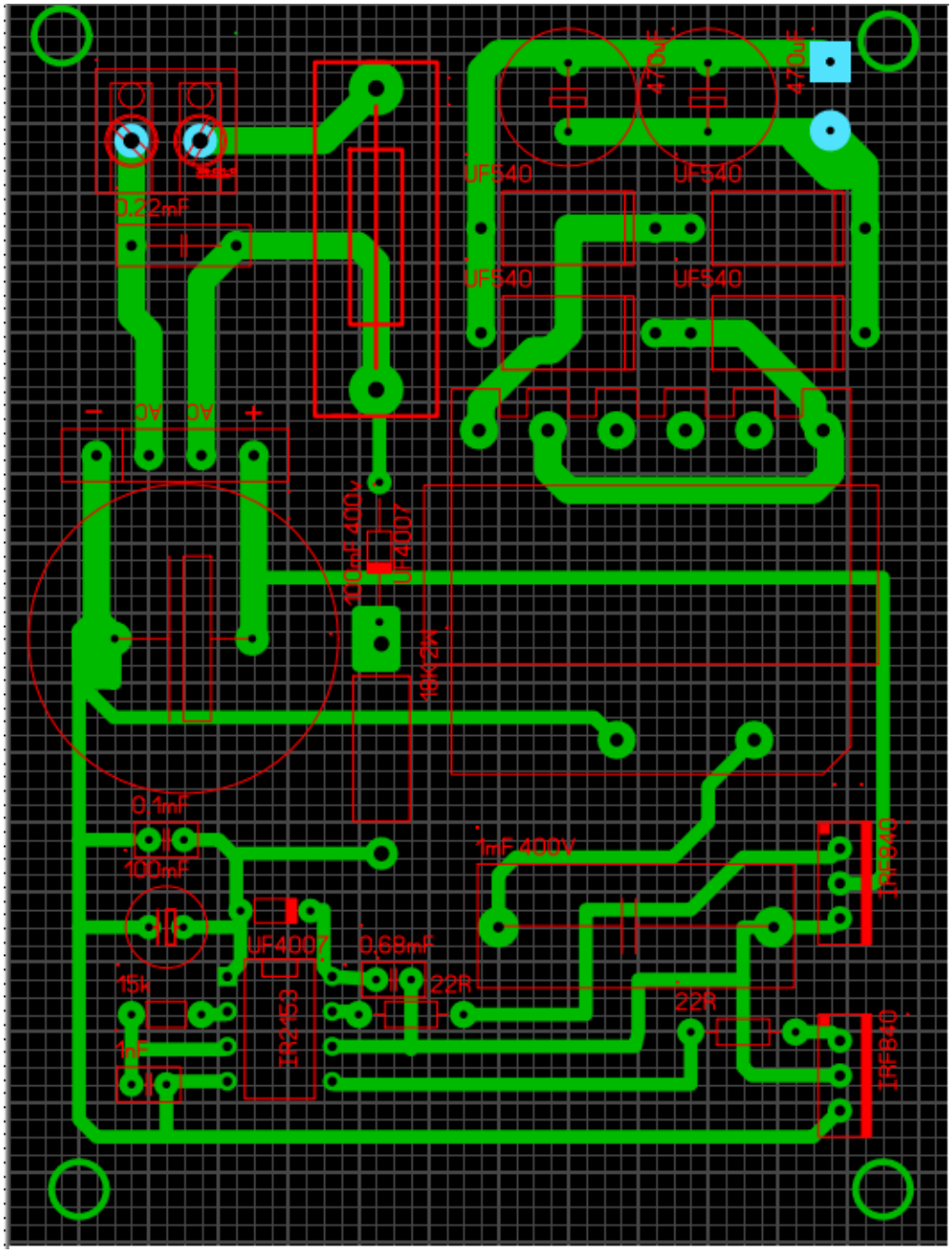


Рисунок 2.15 – Друкована плата блока живлення

2.10 Електрична принципова схема приладу

Схема складається з блока живлення на 12В який живить елементи схеми такі як помпа для зрошення, вентилятори та зволожувач повітря. Від блока живлення напруга йде на лінійний стабілізатор (l7805cv) який стабілізує напругу до 5В необхідних для живлення мікроконтролера, датчиків, мікросхеми годинника реального часу, перетворювача інтерфейсу та LCD екрану.

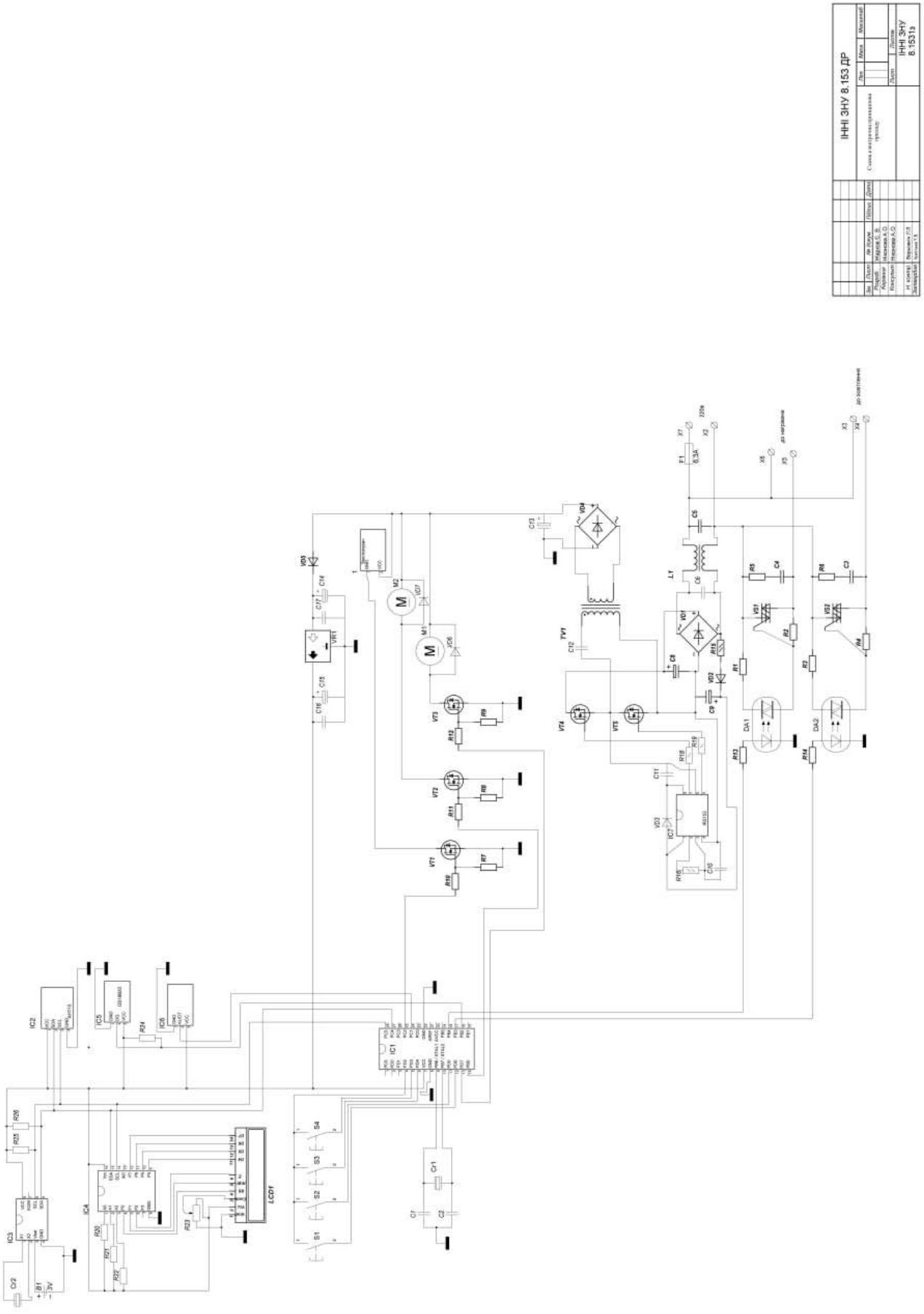


Рисунок 2.16 – Схема електрична принципова приладу

Тактування мікроконтролера ATmega328P (IC1) виконується за допомогою зовнішнього тактового генератора побудованого на кварцевому резонаторі (Cr1) та конденсаторах C1 і C2 номінали яких визначаються виробником МК для конкретної частоти резонатора. Використання кварцового резонатора дозволяє забезпечити високу точність і стабільність тактової частоти. До виводів 27, 28 (шини I2C) мікроконтролера підключені годинник реального часу (IC3), який забезпечує точний відлік часу завдяки часовому кварцевому резонатору (Cr2) навіть за відсутності живлення завдяки батареї B1 годинник продовжує відлік часу, перетворювач інтерфейсу pcf8674 (IC4) який дозволяє керувати LCD екраном 1602 використовуючи лише 2 виводи МК а не 6. До екрану підключено змінний резистор для регулювання контрасту , також на цій шині знаходиться датчик температури та вологості AHT10 (IC2). Для стабільної роботи шини встановлено підтягуючі резистори R25 та R26. Керування пристроєм виконується за допомогую кнопок S1, S2, S3, S4 які підключено до виводів 4, 5, 6, 11та12 мікроконтролера. До виводу 16 мікроконтролера підключено датчик температури IC5 ds18b20 для стабільної роботи якого встановлено підтягуючий резистор R24. Для функціонування моніторінгу вологості грунту до 24 виводу мікроконтролера підключено емнісний датчик вологості грунту. Блок комутації складається з двох частин: високовольтної, яка виконує управління світлом та обігрівом та низьковольтної, яка виконує управління помпою полива рослин, вентиляцією та зволоженням повітря. Високовольтна частина складається з симисторів VS1 та VS2, які безпосередньо комутують змінну напругу та оптопари DA1 та DA2, які управляють симісторами та виконують роль гальванічної розв'язки. Оптопараи підключені до мікроконтролера через струмообмежувальні резистори R13 та R14. Для розвантаження симісторів встановлено снабери, які складаються з резисторів R5, R6 та конденсаторів C3,C4. Низковольтна частина призначена для комутації постійного струму та скаладається з n-канальних транзисторів VT1,VT2 та VT3 які підключені до виводів мікроконтролера 13, 14, 25 через резистори R10, R11, R12. Резистори R7, R8, R9 призначені для розрядження ємності затвора.

2.11 Розробка макету прилада

Для макету зроблено каркас зі сталевого прута розміром 1650x670x800 мм та натягнуто поверх фольгоізол, т.я. він має таки переваги як вологостійкість, легкість, невелика вартість, тримає температуру та частково відбиває світло.

У якості нагрівача обрано лампу розжарювання PAR38 потужністю 100W. У якості джерела світла - LED фіто матриця потужністю 20W. Ультразвуковий генератор холодного туману потужністю 12W використано для зволоження повітря.

Для виготовлення з'єднувачів , тримачів вентиляторів та опори було розроблено макети елементів у середовищі FreeCAD (Рис. 2.17-19) та роздруковано на fdm принтері (Рис. 2.20). Макетування прототипу відбувалося за допомогою безпайкової макетної плати

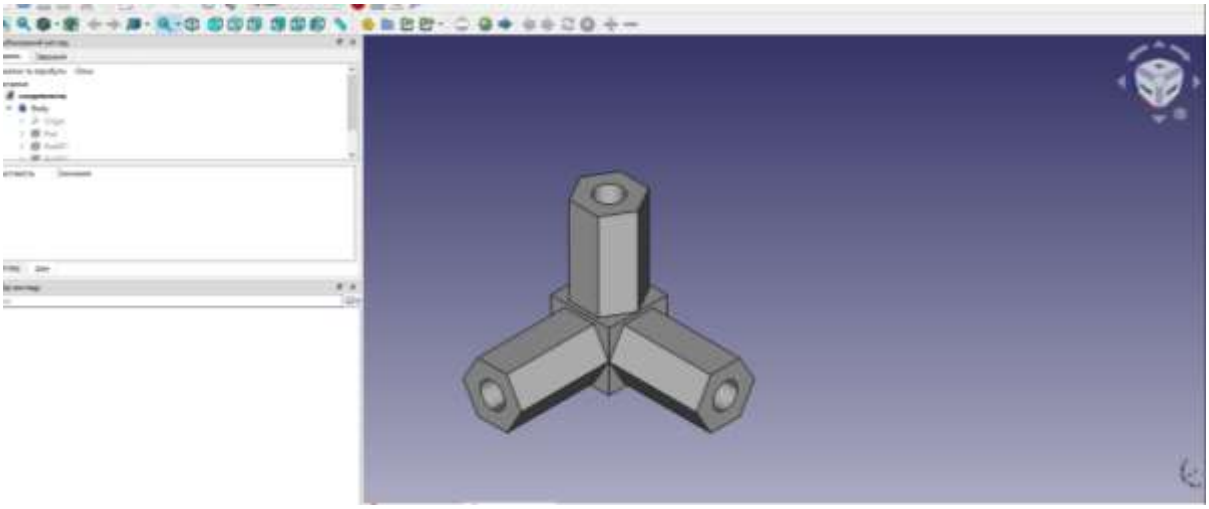


Рисунок 2.17 – Розробка 3d моделі з'єднувача

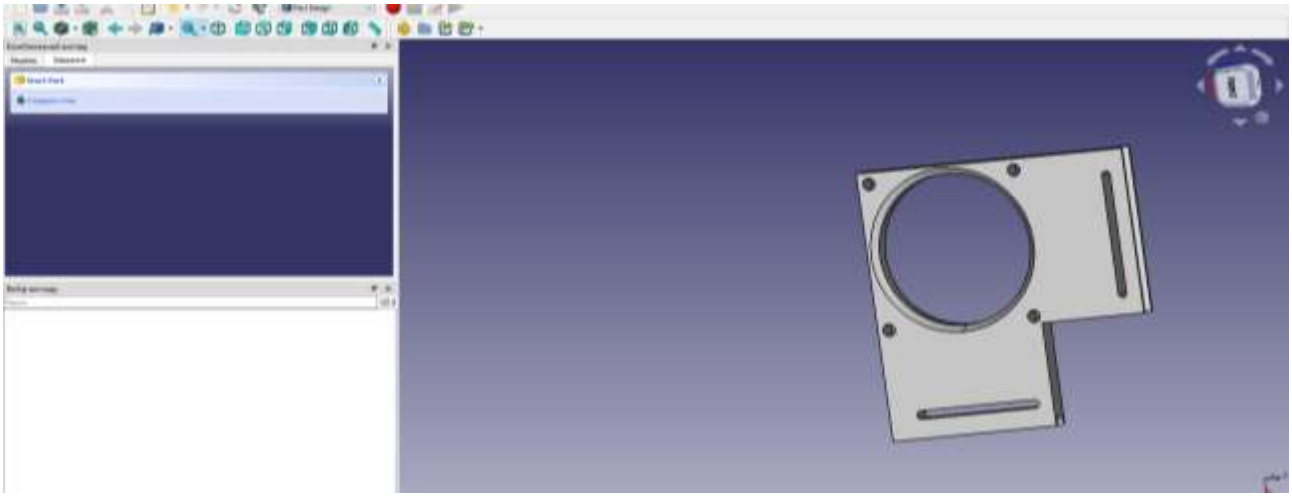


Рисунок 2.18 – Розробка 3d моделі кріплення вентилятора

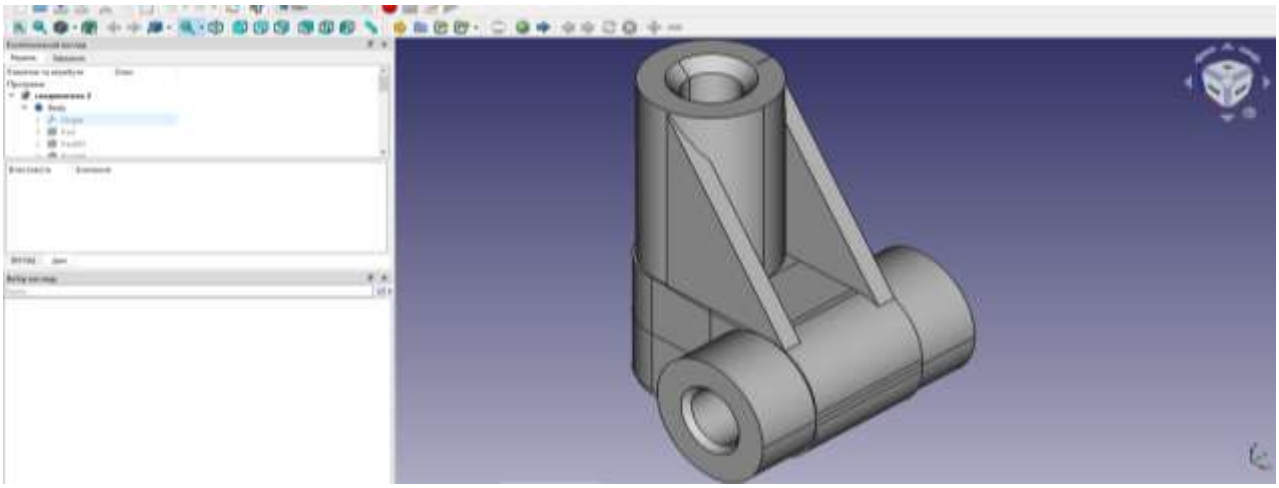


Рисунок 2.19 – Розробка 3d моделі кріплення лампи

Рисунок 2.20 – Зовнішній вигляд розробленої деталі

При макетуванні було випробувано та налаштовано елементи схеми, а саме мікроконтролер та LCD дисплей для коректної роботи приладу(Рис.2.21).

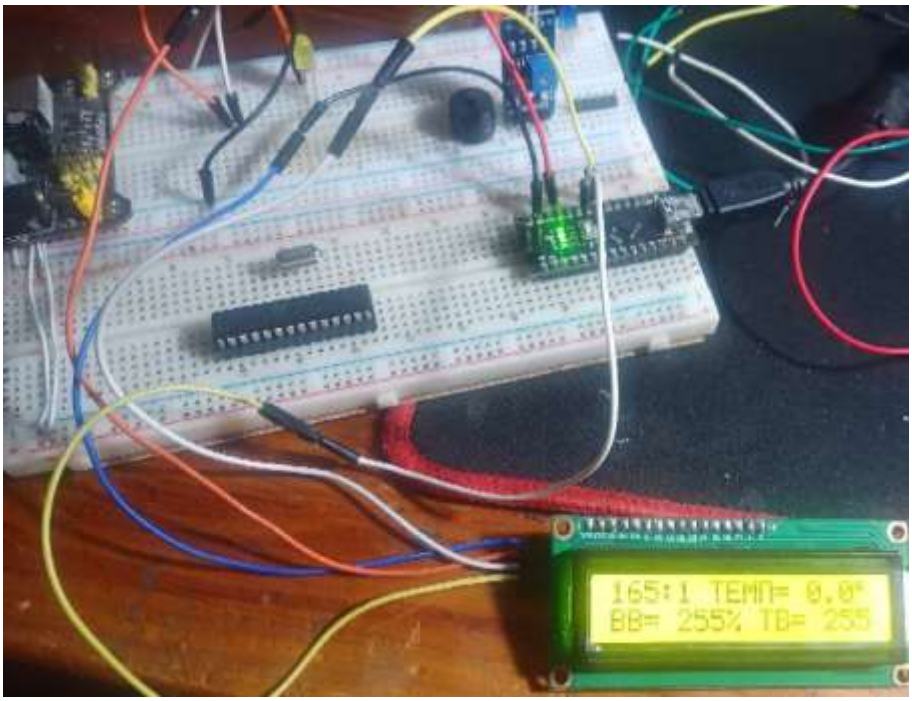


Рисунок 2.21 – Випробування елементів схеми

Виміри електроспоживання приладу у мінімальному режимі роботи показали значення 21Вт (Рис. 2.22).

Рисунок 2.22 - Виміри електроспоживання приладу у мінімальному режимі роботи

За допомогою розробленої системи можна керувати насосом, освітленням, обігрівачем, вентилятором, зволожувачем повітря. Таким чином можна організувати автополив (за часом або показаннями датчика вологості ґрунту), включати і відключати освітлення в потрібний час, забезпечити роботу вентиляції за таймером або при високій температурі, контролювати температуру обігрівача при різних температурних режимах.

Робота приладу відповідає технічним завданням, які оптимально задовольняють поставленим вимогам. Гроубокс задовільно функціонує протягом тривалого часу, тому його використання може бути рекомендовано для вирощування рослин.

3 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

3.1 Обгрунтування вибору температури та вологості АНТ10

Вибір оптимального датчику температури та вологості для системи автоматичного вирощування рослин, найбільшою мірою залежить від вибору задач, які він має виконувати.

Порівняємо ці умови за наступними критеріями:

- за діапазоном вимірвання температури
- за діапазоном вимірюання вологи;
- габаритними розмірами.

До розроблювального приладу пред'являються наступні вимоги, які дозволяють більш плідно вести роботу зі створення даного пристрою:

- простота керування;
- невисока вартість.

При проектування системи доводиться дотримуватись балансу між розмірами і вартістю з одного боку та гнучкістю і продуктивністю з іншого.

Для різних додатків оптимальне співвідношення цих та інших параметрів може відрізнятись дуже сильно. Тому існує великий вибір типів датчиків, що відрізняються параметрами, габаритами, ціною ін..

Проаналізуємо чотири варіанти датчиків (табл.3.1), враховуючи шкалу відносної важливості (табл.3.2).

Таблиця 3.1 - Можливі варіанти вибору датчика

Датчик		Тип інтерфейса
A	HDC1080	I2C
B	DHT22	1-Wire
C	DHT11	1-Wire
D	AHT10	I2C

Таблиця 3.2 - Шкала відносної важливості

Інтенсивність відносної важливості	Визначення
1	рівна важливість
3	помірна перевага
5	сильна перевага
7	значна перевага
9	дуже сильна перевага
2,4,6,8	проміжні судження

Вибір робимо за критеріями, наведеними в таблиці 3.3.

Встановлюємо відносну вагу кожного критерію на основі матриці попарних порівнянь для обраних критеріїв (таблиця 3.3),

У матриці прийняті наступні позначення:

i – номер критерію;

при порівнянні 6-ох критеріїв (табл. 3) *i* = 1, 2, 3, 4, 5, 6;

X_i - локальний пріоритет, тобто відносна вага і-го критерію в глобальному критерії:

$X_i = \frac{\sqrt[6]{\prod_{i=1}^{6} \omega_i}}{\sum_{i=1}^{6} \sqrt[6]{\prod_{i=1}^{6} \omega_i}}$, $\sum$ - сума по стовпці $\sqrt[6]{\prod_{i=1}^{6} \omega_i}$;

Таблиця 3.3 – Попарне порівняння критеріїв

Критерій	1	2	3	4	5	6	$\sqrt[6]{\prod_{i=1}^{6} \omega_i}$	X_i
1. Габарітні розміри	1	1/3	3	1/7	1/5	3	0,664	0,073
2. Швидкодія	3	1	3	1/3	1/7	3	1,042	0,116
3. Точність вимірювання	1/3	1/3	1	1/5	1/7	3	0,460	0,051
4. Вартість	7	3	5	1	1/5	7	2,297	0,254
5.Діапазон вимірювання	5	7	7	5	1	5	4,277	0,473
6. Тип інтерфейсу	1/3	1/5	1/3	1/7	1/5	1	0,293	0,033
$\sum$							9,033	1,00

Порівняння проводимо так: відносна вага кожного критерію самого до себе дорівнює 1. Почнемо, наприклад, з критерію «габарітні розміри»: відносно критерію «вартість» він має значну перевагу (за табл.2 оцінка – 7), тоді в 4-й строці, 1-му стовпчику ставимо 7, а в 1-й строці, 4-му стовпчику ставимо 1/7;

- відносно критерію «швидкодія» він має помірну перевагу (за табл.2 оцінка – 3), тоді в 4-й строці, 2-му стовпчику ставимо 3, а в 2-й строці, 4-му стовпчику ставимо 1/3;

- відносно критерію «точність вимірювання» він має сильну перевагу (за табл.2 оцінка – 5), тоді в 4-й строці, 3-му стовпчику ставимо 5, а в 3-й строці, 4-му стовпчику ставимо 1/5; і т. д. порівнюємо цей критерій з іншими.

Так само порівнючи кожний критерій з іншими, заповнюємо таблицю 3.

Далі в кожній строчці перемножуємо усі 6 значень і беремо з цього добутку корінь 6-го ступеню – так заповнюємо стовпчик $\sqrt[6]{\prod_{i=1}^{6}\omega_i}$; знаходимо суму по цьому стовпчику $\sum$, знаходимо

$$X_i=\frac{\sqrt[6]{\prod_{i=1}^{6}\omega_i}}{\sum_{i=1}^{6}\sqrt[6]{\prod_{i=1}^{6}\omega_i}}$$

для кожної строки і заповнюємо стовпчик X_i.

Далі аналогічно складаємо 6 матриць попарних порівнянь альтернатив стосовно кожного критерію (таблиці 4, 5, 6, 7, 8, 9). Оскільки тепер порівнюються 4 технології по одному критерію, то i = 1, 2, 3, 4;

$X_i=\frac{\sqrt[4]{\prod_{i=1}^{4}\omega_i}}{\sum_{i=1}^{4}\sqrt[4]{\prod_{i=1}^{4}\omega_i}}$; $\sum$ - сума по стовпці $\sqrt[4]{\prod_{i=1}^{4}\omega_i}$.

Таблиця 3.4 - Порівняння альтернатив стосовно критерію «габарітні розміри»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	1/5	3	3	1,16	0,19
B	5	1	7	7	3,96	0,65
C	1/3	1/7	1	1/2	0,39	0,07
D	1/3	1/7	2	1	0,56	0,09
$\sum$					6,07	1,00

Таблиця 3.5 - Порівняння альтернатив стосовно критерію «швидкодія»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	5	1	1/3	1,14	0,21
B	1/5	1	1/5	1/7	0,48	0,09
C	1	5	1	1/3	1,56	0,29
D	3	7	3	1	2,20	0,41
∑					5,37	1,00

Таблиця 3.6 - Порівняння альтернатив стосовно критерію «точність вимірювання»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	5	1/5	1/7	0,61	0,09
B	1/5	1	1/7	1/9	0,24	0,04
C	5	7	1	1/3	1,85	0,29
D	7	9	3	1	3,71	0,58
∑					6,41	1,00

Таблиця 3.7 - Порівняння альтернатив стосовно критерію «вартість»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	5	3	5	0,99	0,21
B	1/5	1	1/3	1/2	0,24	0,05
C	1/3	3	1	3	2,03	0,43
D	1/5	2	1/3	1	1,47	0,31
∑					4,73	1,00

Таблиця 3.8 - Порівняння альтернатив стосовно критерію «діапазон вимірювання»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	5	1/3	1/5	0,76	0,13
B	1/5	1	1/7	1/9	0,24	0,04
C	5	7	1	1/2	2,41	0,41
D	3	9	2	1	2,47	0,42
∑					5,88	1,00

Таблиця 3.9 - Порівняння альтернатив стосовно критерію «тип інтерфейсу»

Датчик	A	B	C	D	$\sqrt[4]{\prod_{i=1}^{4}\omega_i}$	X_i
A	1	3	5	6	3,08	0,56
B	1/3	1	3	4	1,41	0,26
C	1/5	1/3	1	2	0,60	0,11
D	1/6	1/4	1/2	1	0,38	0,07
∑					5,47	1,00

Глобальний пріоритет для кожної альтернативи обчислюється як сума добутків кожного локального пріоритету на його ваговий коефіцієнт. В таблиці 10 строка «вага» - це стовпчик X_i таблиці 3, строка «HDC1080» - це значення X_i таблиць 4 – 9 для датчика А, строка «DHT22» - датчика В і т. д. Глобальний пріоритет для кожної технології розраховуємо так:

- для датчику HDC1080: 0,073х0,19 + 0,116х0,21 + 0,051х0,09 + 0,254х0,21 + 0,473х0,13 + 0,033х0,56 = 0,176 і т.д.

Таблиця 3.10 - Глобальний пріоритет для кожної альтернативи

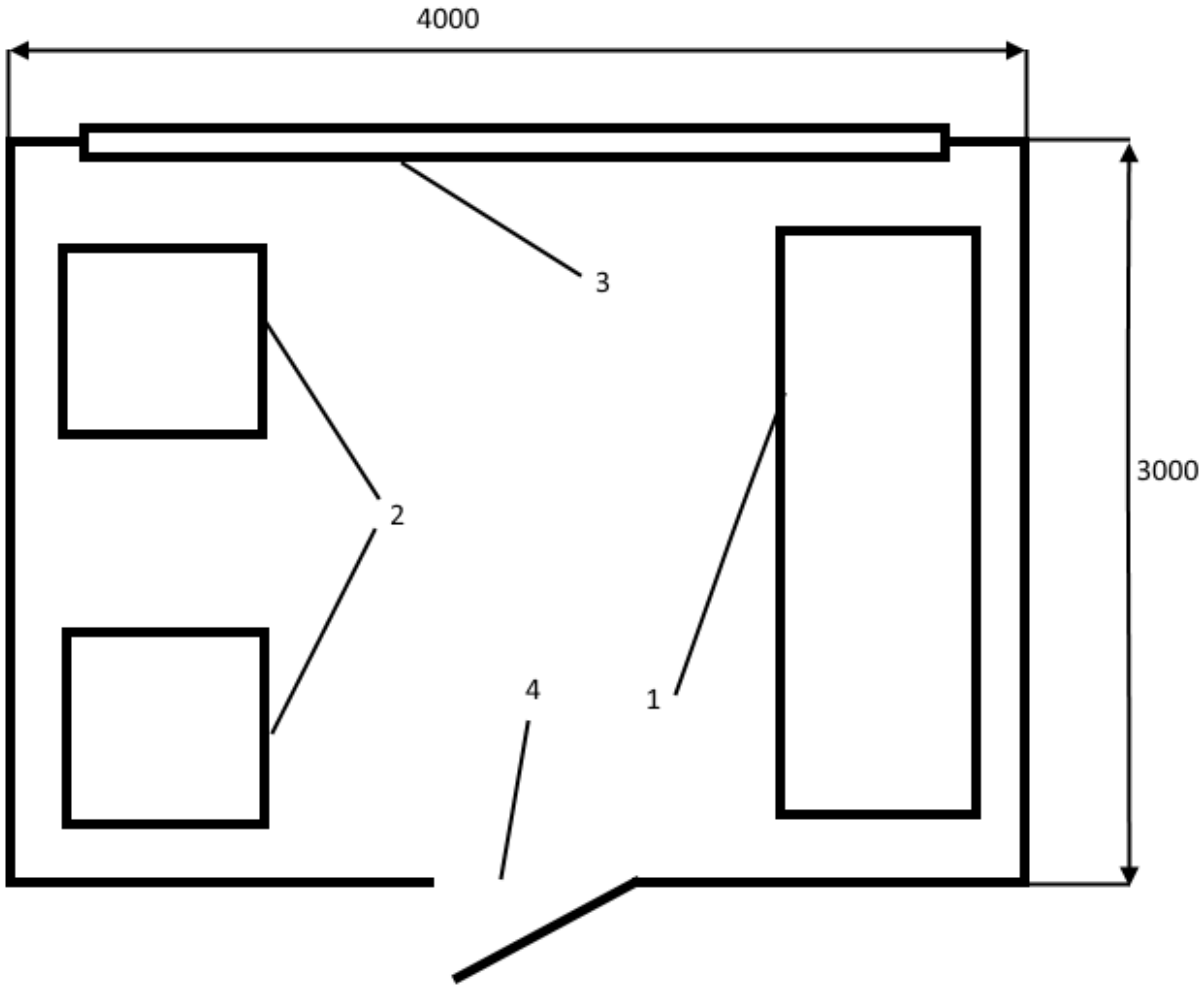
Пріоритети	№1	№2	№3	№4	№5	№6	Глобальний
Вага	0,073	0,116	0,051	0,254	0,473	0,033	
HDC1080	0,19	0,21	0,09	0,21	0,13	0,56	0,176
DHT22	0,65	0,09	0,04	0,05	0,04	0,26	0,100
DHT11	0,07	0,29	0,29	0,43	0,41	0,11	0,360
AHT10	0,09	0,41	0,58	0,31	0,42	0,07	0,364

З порівняння глобальних пріоритетів різних датчиків (табл.3.10) видно, що найбільшим є пріоритет у варіанта датчика температури та вологості AHT10.

4 ОХОРОНА ПРАЦІ ТА ТЕХНОГЕННА БЕЗПЕКА

4.1 Характеристика потенційно небезпечних та шкідливих виробничих факторів

Розробка мікроелектронного пристрою для визначення рівня радіації здійснювалась у лабораторному приміщенні кафедри МЕІС (рис 4.1).



1 – робочий стіл, 2 – робоче місце операторів ПК, 3 – вікно,

4 – вхід у приміщення

Рисунок 4.1 – Схематичне зображення лабораторного приміщення

Параметри приміщення для проведення досліджень вказані у (табл. 4.1).

Таблиця 4.1 – Параметри приміщення

Параметр приміщення	Числове значення
Довжина	4м
Ширина	3м
Висота	3,5м
Площа	12м^2
Об’єм	42м^3

Небезпечні та шкідливі виробничі фактори поділяються на чотири групи:

-фізичні;

-хімічні;

-біологічні;

-психофізичні.

При роботі за ПК найбільш уразливими стають нервова, імунна, зорова, ендокринна, опорно-рухова та репродуктивна системи користувачів. Саме тому комп’ютеризовані робочі місця відносяться до категорії небезпечних для стану здоров’я людини.

Небезпечні та шкідливі фактори, що діють на користувача комп’ютера в процесі роботи, наведені на (рис. 4.2).

Рисунок 4.2 - Небезпечні та шкідливі фактори для користувача ПК

Дія шкідливих факторів призводить до порушення здоров'я. Причини відхилень в здоров'ї користувача ПК наведені у (табл. 4.2).

Таблиця 4.2 - Причини відхилень в здоров'ї користувача ПК

Шкідливий чинник	Захворювання
Незадовільні ергономічні характеристики монітора	- порушення зору
Незадовільні санітарно-гігієнічні умови праці	- захворювання шкіри
Неправильна організація робочого місця	- порушення пов'язані з нервовим та емоційним навантаженням; - кістково-м'язові порушення

Дисплей ПК на електронно-променевій трубці є джерелом електромагнітних випромінювань, які включають:

-радіочастотне;

-рентгенівське;

-ультрафіолетове;

-інфрачервоне випромінювання.

Електромагнітні поля навколо комп'ютера (особливо низькочастотні) негативно впливають на людину. Найбільш чуттєвою до дії електромагнітних випромінювань (ЕМВ) є центральна нервова система.

Вплив електромагнітного випромінювання на нервову систему:

-погіршення пам'яті;

-безсоння;

-депресія та головні болі;

-погіршується сприймання інформації;

-запаморочення;

-"безпричинна" втома.

Випромінювання ПК співпадає з частотою сигналів, які посилають одне одному клітини нашого організму. Клітини виконують найрізноманітніші завдання: передача нервових імпульсів і обробка інформації, транспортування кисню та поживних речовин, синтезу й виділення різних сполук, скорочення м'язових волокон. Таким чином, в механізми внутрішнього спілкування організму людини втручаються штучні сторонні чинники. В результаті цього руйнується інформаційно-керуюча система організму, збивається ритм роботи і, як наслідок, запускаються патологічні процеси.

Електромагнітне випромінювання ослаблює імунну систему, розбалансовує ендокринну та статеву системи, що призводить до передчасного старіння.

Особливо уразливі до випромінювання моніторів ПК вагітні жінки. Ризик появи дітей з уродженими хибами збільшується в 25 раз, в 3,5 рази вища імовірність викидів.

Електростатичне поле, яке створюють монітори на електронно-променевій трубці може негативно вплинути на самопочуття, нервову і судинну системи, також воно може бути причиною бронхо-легеневих та шкірних захворювань.

Накопичена статична електрика, зокрема, на екрані монітора притягує пил, бруд та інші частини присутні в повітрі. При чому електризується не тільки екран, а і повітря на робочому місці, а також одяг користувача, якщо він з синтетичного та шерстяного матеріалу.

При напруженій роботі за екраном монітора зменшується частота кліпання очей, що призводить до висихання та викривлення роговиці ока, погіршення зору.

Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98.

При виготовленні апарату для реєстрації радіоактивного випромінювання найбільш небезпечними, з точки зору охорони праці і техніки безпеки, операціями, що виконуються у лабораторному приміщенні є:

-пайка;

-виготовлення друкованих схем;

-збірка.

При виготовленні приладу були використані припій ПОС-61 (ГОСТ 21930-76) і каніфоль світла (ГОСТ 797-64). При пайці і залуженні олов'яно-свинцевими припоями утворюються пари свинцю.

Свинець негативно впливає на організм, особливо на стан нервової і серцево-судинної системи, викликає ряд хворобливих явищ шлунково-кишкового тракту, що призводить до професійного отруєння.

При систематичній роботі з припоями, що містять свинець, необхідна припливно-витяжна вентиляція, а на робочому місці повинен бути встановлений відсмоктувач. Пайку і лудіння рекомендується проводити в спеціальних витяжних шафах.

Склад припою марки ПОС-61 і допустимих домішок, а також МДК цих матеріалів зазначені в таблиці 3.3.

Таблиця 4.3 - Склад і МДК компонентів припою ПОС-61

Компонент	Вміст	МДК
Олово	60…62 %	0,05 мг/м3
Вісмут	0,1 %	0,5 мг/м3
Миш'як	0,05 %	0,3 мг/м3
Залізо	до 0,02 %	0,004 мг/м3
Нікель	до 0,02 %	0,001мг/м3
Сірка	до 0,02 %	6 мг/м3
Цинк	до 0,002%	0,25 мг/м3
Алюміній	до 0,002%	2 мг/м3
Мідь	до 0,05 %	0,002 мг/м3
Свинець	Інше	0,0003 мг/м3

При роботі з олов'яно-свинцевими припаями потрібно строго виконувати передбачені правила виробничої та особистої гігієни:

-категорично забороняється приймати їжу і курити в приміщеннях, де проводиться паяння;

-перед обідньою перервою і після роботи необхідно обмивати руки 1% розчином соди і ретельно мити їх теплою водою;

-після закінчення роботи слід приймати душ;

-очищення місць пайки від залишків флюсу робити спиртом;

-спецодяг слід регулярно прати і зберігати на виробництві;

-медичний огляд осіб, які працюють з припоями, повинен проводитися не рідше одного разу на рік.

Друковані плати сучасного типу можуть бути виготовлені за однією з двох технологій – адитивним чи субтрактивним способом. У першому випадку провідний малюнок на матеріалі формується за допомогою процедури хімічного міднення.

Процес виробництва друкованих плат складається з таких етапів:

1.Виробництво заготівлі для платі;

2.Подальша обробка заготовки;

3.Здійснення монтажу всіх необхідних елементів плати;

4.Проведення планового тестування працездатності [13].

При нанесенні малюнка струмопровідними фарбами з подальшим гальванічним нарощуванням металу велику шкідливість для організму працюючих представляють різні речовини:

розчинники,

відновники,

наповнювачі, що входять до складу струмопровідних фарб.

Найбільш шкідливо впливають хлорорганічні розчинники (дихлоретан, хлороформ та ін.), ароматичні сполуки (бензол, толуол, ксилол та ін.), фенолальдегідні смоли, формальдегід та інші речовини.

Під час роботи з струмопровідними фарбами велику роль відіграє механізація і автоматизація технологічного процесу, слід виключити можливість безпосереднього контакту працівників з фарбами.

Робоче місце повинно бути обладнано вентиляцією, а працюючий зобов'язаний дотримуватися заходів виробничої і особистої гігієни.

При електролітичному методі нарощування металу необхідно дотримуватися санітарних вимог і правил з техніки безпеки, прийнятих для гальванічних цехів.

При пайці друкованих схем методом занурення в розплавлений припій ПОС-61 пари свинцю можуть забруднювати повітря.

При ручному способі занурення плат можливі опіки працюючих краплями розплавленого припою, тому процес занурення плат повинен бути механізований. Для підвищення рівня безпеки працівників ванни з розплавленим припоєм повинні мати кришки та пристрої для відсмоктування повітря з ванн.

Процес залуження кінців дротів супроводжується виділенням парів свинцю і флюсів. Тому приміщення в яких проводиться залуження кінців дротів повинні бути обладнаними працюючими місцевими вентиляціями.

Ізоляцію монтажних дротів у ряді випадків виконують різними клеями, а постійне забруднення поверхні шкіри клеями може викликати появу шкірних захворювань. Тому для попередження цих захворювань рекомендується користуватися спеціальними захисними пастами.

Таким чином, на дільниці складання і монтажу на працюючих можуть впливати:

-пари свинцю і флюсів (при пайці);

-пари та дрібний пил металів, що сполучаються;

-газоподібні речовини;

-дим і пари, що утворюються при приготуванні ізоляції проводів.

Тому складальні і монтажні місця повинні бути обладнані ефективною, раціонально влаштованою припливно-витяжною механічною вентиляцією.

4.2 Заходи з поліпшення умов праці. Виробнича санітарія

Вимоги до робочого місця та розташування всіх його елементів прописано в ДНАОП 0.00-1.31-99 "Правила охорони праці під час експлуатації електронно-обчислювальних машин ".

Згідно з вимогам облаштування робочих місць, обладнаних відеотерміналами, повинні забезпечуватись:

-належні умови освітлення приміщення і робочого місця;

-відсутність відблисків;

-оптимальні параметри мікроклімату;

-належні ергономічні характеристики основних елементів робочого місця, з врахуванням небезпечних і шкідливих факторів, які були розглянуті раніше.

Площа приміщень для роботи з відеодисплейним и терміналами розраховується таким чином, щоб площа на одне робоче місце, обладнане відеотерміналом становила не менше 6,0м2, а об'єм на одне робоче місце – не менше 20,0 м3.

Робочі місця мають бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м, між собою на відстані не менше 1,5 м.

Відносно вікон робоче місце доцільно розташовувати таким чином, щоб уникнути попадання в очі прямого світла. Тому місця розміщують так, щоб природне світло падало на нього збоку, переважно зліва.

Джерела освітлення рекомендується розташовувати з обох боків екрану паралельно напрямку погляду.

Під час розробки комбінованої транзисторної структури, яка проводилась у лабораторному приміщенні кафедри ФБМЕ, всі необхідні вищеперераховані вимоги виконувались.

Робочі місця операторів ПК розташовані на відстані 2 метрів від вікна, та 1 метра від стіни, таким чином, що світло з вікна падає на робоче місце збоку.

У приміщенні, площа якого 12 м2 та об'єм 42 м3, знаходиться два робочих місця з комп'ютером, що відповідає санітарнім умовам. Монітор розміщений таким чином, що він знаходиться на оптимальній відстані від очей користувача, 50-60 см.

Виробнича санітарія - це система організаційних заходів і технічних засобів, що запобігають або зменшують вплив на працюючих шкідливих виробничих факторів, які в певних умовах можуть привести до травм або професійних захворювань.

Основною метою є зменшення або повне усунення впливу несприятливих 1 шкідливих виробничих факторів на організм людини.

Оскільки головним у діяльності з охорони праці є профілактика травматизму, заходи щодо поліпшення умов праці й побуту працюючих дозволяють не тільки знизити виробничий травматизм, професійну й загальну захворюваність, а й сприяють підвищенню продуктивності і якості праці [13].

На робочому місці необхідно використовувати правильно спроектоване та безпечне у використанні виробниче освітлення, це допоможе знизити втомлюваність, покращить умови праці.

Недостатня або надмірна освітленість, нерівномірність освітлення в полі зору втомлює очі, призводить до зниження продуктивності праці; при цьому зростає потенційна небезпека помилкових дій і нещасних випадків.

Надмірна яскравість джерел світла може спричинити головний біль, різь в очах, розлад гостроти зору; світлові відблиски — тимчасове засліплення.

Освітлення виробничих приміщень характеризується кількісними та якісними показниками.

До основних кількісних показників відносяться:

-світловий потік;

-сила світла;

-яскравість;

-освітленість.

До основних якісних показників зорових умов роботи можна віднести: контраст між об'єктом та фоном, видимість.

Для створення сприятливих умов зорової роботи виробниче освітлення повинно відповідати наступним вимогам:

-створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

-не повинно бути засліплюючої дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

-забезпечити достатню рівномірність та постійність рівня освітленості у виробничих приміщеннях, щоб уникнути частої переадаптації органів зору;

-не створювати на робочій поверхні різких та глибоких тіней (особливо рухомих);

-повинен бути достатній, для розрізнення деталей, контраст поверхонь, що освітлюються;

-не створювати небезпечних та шкідливих виробничих факторів (шум, теплові випромінювання, небезпечне ураження струмом, пожежо-та вибухонебезпека світильників);

-повинно бути надійним і простим в експлуатації, економічним та естетичним.

Залежно від джерела світла виробниче освітлення може бути природним, штучним і суміщеним, при якому недостатнє за нормами природне освітлення доповнюється штучним.

Недостатність освітлення приводить до напруги зору, послаблення уваги, передчасної стомленості.

Неправильне освітлення може привести до нещасного випадку або профзахворювань, тому настільки важливий правильний розрахунок освітленості.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери: при виконанні зорових робіт середньої точності – 200 лк загального й 300лк для комбінованого освітлення.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури й зниження відносної вологості в приміщенні.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.36.042-99 "Санітарні норми мікроклімату виробничих приміщень" установлені величини параметрів мікроклімату, що створюють комфортні умови.

Ці норми встановлюються залежно від пори року, характеру трудового процесу й характеру виробничого приміщення (таблиця 4.4).

Таблиця 4.4 – Оптимальні параметри мікроклімату приміщень, для категорії робіт 1А

Період року	Параметр мікроклімату	Величина
Холодний	Температу раповітря в приміщенні	22...24°С
	Відносна вологість	40...60%
	Швидкість руху повітря	до 0,1м/с
Теплий	Температура повітря в приміщенні	23...25°С
	Відносна вологість	40...60%
	Швидкість руху повітря	0,1...0,2м/с

Відповідно до норм подачі свіжого повітря в приміщення, де розташовані комп'ютери в даній лабораторії необхідно подавати 30 м3/год, на одну людину

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року й доби, чергування праці й відпочинку), так і технічні засоби (вентиляція, кондиціонування повітря, опалювальна система).

Шум погіршує умови праці, роблячи шкідливу дію на організм людини, при роботі в умовах тривалого шумового впливу робітники відчувають:

-дратівливість;

-головні болі;

-запаморочення;

-знижується концентрація уваги;

-підвищену стомлюваність;

-зниження апетиту;

-біль у вухах і т.д.

Рівень шуму в даному приміщенні становить 50дБ, що не перевищує норму. Для зниження рівня шуму стіни й стеля приміщення, де встановлені комп'ютери, облицьовані звуковбирними матеріалами.

4.3 Електробезпека

Оскільки в приміщенні знаходиться електроустаткування, основні заходи щодо техніки безпеки повинні стосуватися безпечної експлуатації ПЕОМ, і здійснюватись відповідно до НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів».

Приміщення, в якому знаходиться робоче місце, відноситься до приміщень без підвищеної небезпеки.

Ураження електричним струмом може статись через:

-пошкодження ізоляції;

-випадковий дотик до струмоведучих деталей;

-замикання в результаті аварії;

-статичну напругу.

Характеристика мережі у приміщенні: 220/380В, 50Гц, 3-фазна чотирьох провідна з глухо заземленою нейтраллю.

Електричний струм, проходячи через організм людини, спричиняє термічну, електролітичну та біологічну дії.

Термічна дія струму виявляється в опіках окремих ділянок тіла, ураженні внаслідок високої температури кровоносних судин, нервових клітин, серця, мозку, що призводить до серйозних функціональних розладів.

Електролітична дія струму виявляється в розкладанні органічних рідин, в тому числі крові, що призводить до значних порушень їх фізико-хімічного складу.

Біологічна дія струму виявляється у подразненні й збудженні живої тканини організму, що супроводжується мимовільним скороченням м'язів [13].

Найчастіше нещасні випадки відбуваються через низький рівень організації робіт або грубе порушень правил, у тому числі:

-безпосереднього дотику до відкритих струмоведучих частин і дротів;

-дотику до струмоведучих частин, ізоляція яких ушкоджена;

-дотику до металевих частин устаткування, що випадково під напругою;

- дотик до струмоведучих частин за допомогою предметів з низьким опором ізоляції;

-відсутності або порушення захисного заземлення;

-помилкової подачі напруги під час ремонтів або оглядів;

-впливу електричного струму через дугу;

-впливу крокової напруги й ін.

Для усунення небезпеки ураження електричним струмом в лабораторному приміщенні під час аварійного режиму використовується захисне заземлення.

Заземлення – це спеціальне електричне сполучення із землею або її еквівалентом струмопровідних елементів обладнання, які не повинні перебувати під напругою, але в процесі експлуатації можуть опинитися під напругою, наприклад, у разі пошкодження ізоляції, дефектів комутаційних апаратів, в аварійних випадках тощо.

4.3.1 Розрахунок захисного заземлення

Заземленню підлягають вимірювальні установки, напруга живлення яких 220В.

У якості заземлювача візьмемо сталеві вертикальні стержні довжиною l = 2 м, діаметром d = 0,03 м, діаметр сполучної смуги b = 0,03 м.

Контур заземлення розташовано на горизонтальному майданчику біля корпусу академії.

Грунт - суглинок.

Допустимий опір заземлюючого пристрою (Rн) дорівнює 4 Ом. Визначимо питомий електричний опір грунту (суглинку), ρ якого дорівнює 100 Ом·м. Розрахуємо опір розтікання струму одного вертикального стержня:

$$R_в = \frac{\rho}{2\pi l} \cdot \ln\frac{4 \cdot l}{d} = \frac{100}{2 \cdot 3{,}14 \cdot 2} \ln\frac{4 \cdot 2}{0{,}03} = 44{,}46 \text{ Ом} \qquad (4.1)$$

Приймемо число заземлювачів:

$$R_в = \frac{\rho}{2\pi l} \cdot \ln\frac{4 \cdot l}{d} = \frac{100}{2 \cdot 3{,}14 \cdot 2} \ln\frac{4 \cdot 2}{0{,}03} = 44{,}46 \text{ Ом} \quad (4.2)$$

Визначимо довжину горизонтальної смуги (по контуру)

$$l_г = a \cdot n = 2 \cdot 12 = 24 \text{ м} \quad (4.3)$$

де a - відстань між вертикальними електродами (a = 2 м)

Опір горизонтальної смуги:

$$R_{2c} = \frac{\rho}{\pi l_г} \ln\frac{4 \cdot l_г}{b} = \frac{100}{3{,}14 \cdot 24} \ln\frac{4 \cdot 24}{0{,}03} = 10{,}7 \text{ Ом} \quad (4.4)$$

Коефіцієнт екранування для вертикального заземлювача і для горизонтальної смуги:

$$\eta_в = \frac{0{,}68 + 0{,}56}{2} = 0{,}62 \quad (4.5)$$

$$\eta_г = \frac{0{,}34 + 0{,}4}{2} = 0{,}37 \quad (4.6)$$

Опір групи вертикальних заземлювачів:

$$R_{гр} = \frac{R_{вз}}{n \cdot \eta_в} = \frac{44{,}46}{12 \cdot 0{,}62} = 5{,}97 \text{ Ом} \quad (4.7)$$

Опір горизонтальної смуги з врахуванням екранування:

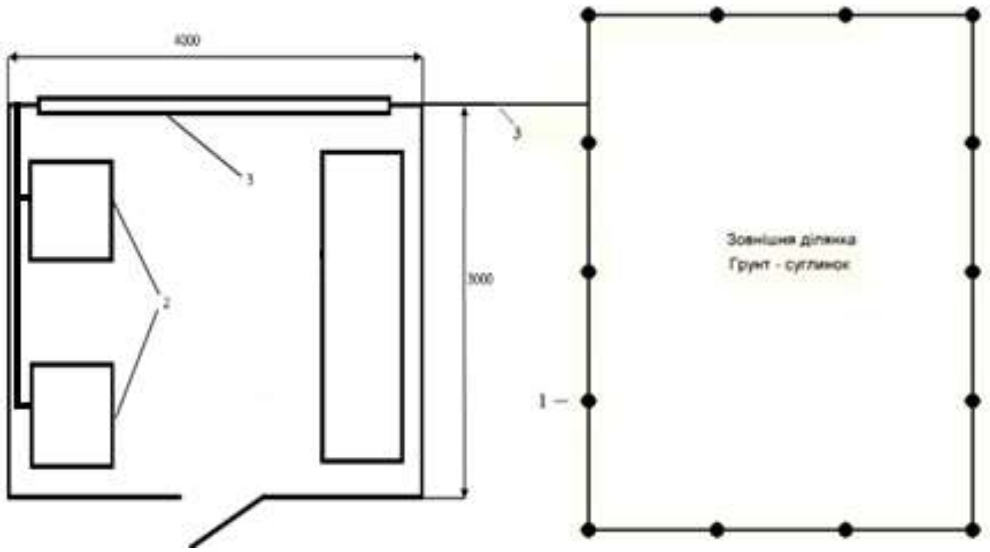
$$R_г = \frac{R_{го}}{\eta_г} = \frac{10{,}7}{0{,}37} = 28{,}91 \text{ Ом} \quad (4.8)$$

Опір заземлювача в цілому:

$$R_з = \frac{R_{гр} \cdot R_г}{R_{гр} + R_г} = \frac{5{,}97 \cdot 28{,}91}{5{,}97 + 28{,}91} = 4{,}94 \text{ Ом} \quad (4.9)$$

Як видно, опір заземлювача перевищує нормативне значення (4 Ом), тому збільшимо кількість заземлювачів до n =14,тоді опір заземлювача в цілому складе Rз = 3,97<Rн.

Число заземлювачів дорівнює 14, які розміщені по контуру зовнішньої ділянки, як показано на рисунку 4.3.



1 – заземлювачі; 2 - електроустаткування що заземлюється; 3 - сполучна смуга внутрішній контур заземлення.

Рисунок 3.3 — Розміщення заземлювачів на зовнішній ділянці.

Таким чином, для виконання захисного заземлення використовується 14 вертикальних стержнів з такими характеристиками:

діаметр - 0,03 м;

довжина - 2 м;

опір стержня - 44,46 Ом;

опір сполучної смуги 10,7 Ом;

опір заземлювачів в цілому - 3,97 Ом.

Такі характеристики заземлення забезпечують безпечну роботу на установках лабораторії.

4.4 Пожежна безпека. Техногенна безпека

Пожежа в лабораторному приміщенні, де розташовано робоче місце, може виникнути при взаємодії горючих речовин і джерел запалювання.

Горючими речовинами в лабораторії можуть стати:

матеріали меблів,

пластмасові корпуси техніки,

шнури тощо.

Джерелами запалювання можуть бути:

Електронні схеми комп'ютерів,

Пристрої електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри, здатні спричинити займання горючих матеріалів.

Клас пожежі Е, категорія приміщення Д.

Для ліквідації пожежі в даному приміщенні немає необхідності влаштування системи автоматичного пожежогасіння.

Приміщення оснащене переносним вуглекислотним вогнегасником типу ВВК-3,5 - 1шт.

Облицювання стін та стелі приміщення зроблене з негорючих матеріалів.

Коридори будівлі, в якому знаходиться дане приміщення, оснащені стендами з планом евакуації під час пожежі (рис. 3.4), на стіні є ящик пожежним стволом і пожежним рукавом.

Дана будівля відносяться до другої категорії по блискавко захисту, захист будівлі від прямих ударів блискавки здійснюється за допомогою стрижневих блискавковідводів.

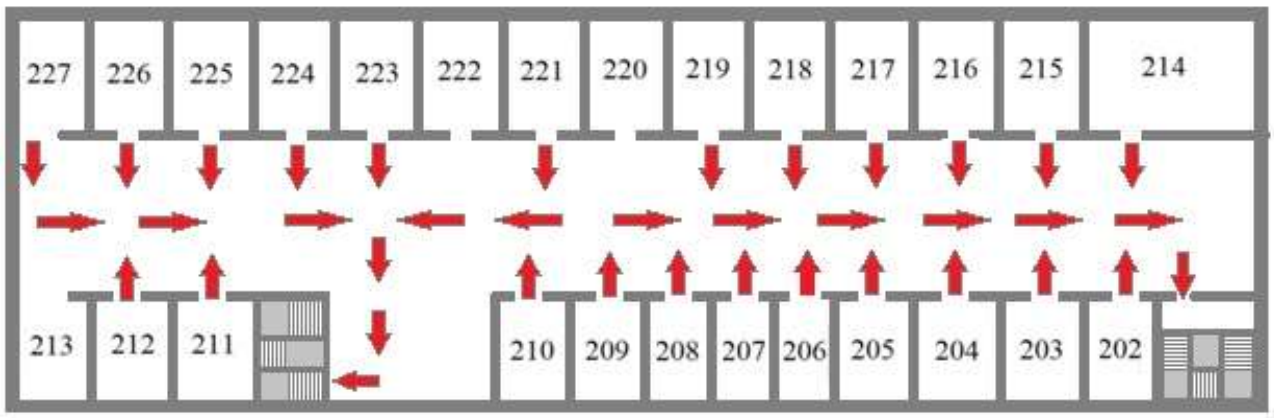


Рисунок 4.4– План евакуації при пожежі

Висновки та рекомендації

1. Проведено аналіз існуючих аналогів систем автоматичного зрошування рослин, виявлено складові системи та параметри які необхідно контролювати для підтримки в Grow box рослини на всіх етапах росту і цвітіння. Найбільшою перевагою використання приладу є те, що рослини захищені від шкідників, паразитів, бактерій та не потребують використання хімікати, необхідних для знешкодження шкідників.

2 Для дотримання компромісу між розмірами та вартістю схеми управління з одного боку та гнучкістю та продуктивністю з іншого, обрано мікроконтролер Atmega328 – найпоширеніший, здатний працювати з усіма необхідними портами, економічний, надійний та має достатню пам'ять.

3. Проведено порівняльний аналіз датчиків та техніко-економічне обґрунтування в результаті якого обрано датчик температури та вологості AHT10, датчик температури DS18B20 та ємнісний датчик вологості грунту .

4. Для передачі посилених по потужності сигналів по зонах оповіщення, а також для забезпечення контролю відсутності короткого замикання чи обриву в лініях зон оповіщення розроблено блок комутації, який складається з двох блоків на основі симісторів BTA16-600B та польових транзисторів 70to3gh. Функціонування схеми перевірено в програмі Proteus.

5. Блоку живлення розроблено на основі мікросхеми IR2153. Перевага цієї схеми в тому, що схема має достатньо високий ККД та невеликі розміри. Схема живиться від напруги 220 вольт, має на вході фільтр, який складається з дроселя і двох плівкових конденсаторів розрахованих на напругу не менше 250 - 300 Вольт ємністю від 0,1 до 0,33 мкФ

6. Розробка електричної схеми приладу та симуляція її елементів проведено у програмі Proteus. Трасування друкованих плат блока живлення та основної плати - у середовищі Sprint_Layout.

7. Для виготовлення з’єднувачів , тримачів вентиляторів та опори було розроблено макети елементів у середовищі FreeCAD та роздруковано на fdm принтері. Тестування розробленого прототипу приладу дало очікувані задовільні

результати, прилад функціонує протягом тривалого часу, тому його використання може бути рекомендовано для процесу вирощування рослин.

ПЕРЕЛІК ПОСИЛАНЬ

1. Технологія виробництва продукції рослинництва : навч. посіб. Ч.1 / [Мельник С.І., Муляр О.Д., Кочубей М.Й., Іванцов П.Д.]. – К. : Аграрна освіта, 2010. – 282 с.
   ISBN 978-966-7906-89-4
2. Тихонов Є. С. Виконавчий пристрій для переміщення деталей на виробничій лінії.: матеріали конференції «Автоматизовані системи та комп'ютеризовані технології радіоелектронного приладобудування», 7-9 квітня 2020 року, ХНУРЕ, м. Харків, Україна, С. 15 – 16
3. Верьовкін Л.Л., Світанько М.В., Кісельов Є.М., Хрипко С.Л. Цифрова схемотехніка: Підручник. Запоріжжя : ЗДІА, 2016. 214 с. 77
4. Фізичні основи електроніки: курс лекцій [Електронний ресурс]: навч. посіб. для студ. спеціальності 171 «Електроніка» / КПІ ім. Ігоря Сікорського; уклад.: К.С. Дрозденко, Електронні текстові данні (1 файл: 8,58 Мбайт). Київ: КПІ ім. Ігоря Сікорського, 2021. 153 с.
5. Сучасні мікроконтролери. Теорія і практика використання стандартних модулів Arduino: [навч. посіб. для студентів ВНЗ] / А. А. Зорі, В. П. Тарасюк, О. А. Штепа ; Держ. ВНЗ «Донец. нац. техн. ун-т». — Покровськ (Донец. обл.): ДонНТУ, 2017. — 281 с. : іл., табл. — ISBN 978-966-377-209-7.
6. Лисенков М. О. Мікроконтролери в приладах і пристроях: підруч. для студ. техн. спец. вищ. навч. закл. / М. О. Лисенков, І. І. Ключник ; МОН України, Харк. нац. ун-т радіоелектроніки. — Харків: ХНУРЕ, 2014. — 368 с. : іл. — ISBN 978-966-659-203-6
7. Проектування мікропроцесорних систем: Проектування мікропроцесорних систем на базі AVR-мікроконтролерів: Периферійні модулі AVR-мікроконтролерів: Навчальний посібник для студентів напряму підготовки 6050201 «Системна інженерія» кафедри Автоматики та управління у технічних системах / Укл.: А.О. Новацький– К: НТУУ „КПІ”, 2012– 470с
8. Схемотехніка електронних систем: у 3 кн. Кн. 3. Мікро- процесори та мікроконтролери: Підручник / В. І. Бойко,2004р

9. Основи електроніки: функціональні елементи та їх застосування. Підручник для студентів неелектротехнічних спеціальностей вищих навчальних закладів. Львів: «Новий Світ-2000»; «Магнолія плюс».2003. 208 с.
http://dspace.wunu.edu.ua/bitstream/316497/8500/1/fkit_kki_dke_ksm_LEK.pdf
10. Будіщев М.С. «Електротехніка, електроніка та мікропроцесорна техніка», Львів, «Афіша», 2001, 424с.
http://194.44.152.155/elib/local/sk657185.pdf
11. Мікроконтролерні пристрої: навч. посіб. для студ. спец. «Мікро- та нано-електроніка» / О. С. Тонкошкур, І. В. Гомілко, О. В. Коваленко ; Дніпропетровський нац. ун-т ім. О. Гончара. — Д. : Вид-во ДНУ, 2011. — 264 с.
12. Основи схемотехніки електронних систем: Підручник [Текст]/ [Бойко В.І., Гуржій А.М., Жуйков В.Я. та ін.].- К.:Вища шк., 2004-527 с.
13. Кожемякін Г.Б., Рижков В.Г., Бєлоконь К.В. Охорона праці та техногенна безпека: методичні вказівки до виконання розділу магістерських робіт для студентів ЗДІА всіх спеціальностей денної та заочної форм навчання. – Запоріжжя: ЗДІА, 2012. – 48 с.

ДодатокА

Позн	Найменування	Кільк	Примітка
	Діодні мости		
VD1	*1а 1000в*	1	
VD4	*HER308*	1	
	Діоди		
VD2, VD6, VD7	*HER308*	3	
VD3	*1N4007*	1	
	Резистори		
R1,R3	*ОМЛТ-0,125-380Ом±5%*	2	
R2,R4,R13,R14	*ОМЛТ-0,125-330Ом±5%*	4	
R5-R6	*ОМЛТ-0,125-39Ом±5%*	2	
R7 – R9, R17-R19	*ОМЛТ-0,125-100кОм±5%*	6	
R10 – R12	*ОМЛТ-0,125-100Ом±5%*	3	
R15	*ОМЛТ-2-24кОм±5%*	1	
R16	*ОМЛТ-0,125-1кОм±5%*	1	
R20-R22	*ОМЛТ-0,125-100кОм±5%*	3	
R24-R26	*ОМЛТ-0,125-4,7кОм±5%*	3	
	Резистори змінні		
R23	*3296W-1-10310 кОм*	1	
	Конденсатори керамічні		
C1-C2	*КМ-5 50В 22нФ*	2	
C10	*КМ-5 50В 1нФ*	1	
C11	*КМ-5 50В 220нФ*	1	
C16-C17	*КМ-5 50В 100нФ*	2	

					ІННІ ЗНУ 8.153 ДР

Позн	Найменування	Кільк	Примітка
	Конденсатори плівкові		
C3-C7	*400В 100нФ*	6	
C12	*400В 200нФ*	1	
	Конденсатори електролітичні		
C8	*400в 47мкФ*	1	
C9	*25в 100мкФ*	1	
C13	*35в 1000мкф*	1	
C14	*35в 220мкФ*	1	
C15	*16в 47мкФ*	1	
	Батареї		
B1	*Cr2032*	1	
	Транзистори		
VT1-VT3	*TO-252 - AP70T03GH*	3	
VT4-VT5	*TO-220 - IRF840*	2	
	Симістори		
VS1-VS2	*TO-220-BTA16-600B*	2	
	Оптопари		
DA1-DA2	*DIP-6 - MOC3023*	2	
	Дроселі		
L1	*Дроссель UU10mH синфазний*	1	
TV1	Імпульсний трансформатор	1	

					ІННІ ЗНУ 8.153 ДР	

Позн	Найменування	Кільк	Примітка
	Стабілізатори		
VR1	*TO-220-l7805cv*	1	
	Кварцеві резонатори		
Cr1	*HC-49E - 16MHz*	1	
Cr2	*HF26 - 32768 Гц*	1	
	Мікросхеми		
IC1	*ATmega328P-PU*	1	
IC2	*AHT10*	1	
IC3	*DIP-8 - Ds1307*	1	
IC4	*pcf8674*	1	
IC5	*Ds18b20*	1	
IC6	Датчик волгості грунту	1	
IC7	*IR2153*	1	
	Дисплеї		
LCD1	*LCD 1602*	1	

					ІННІ ЗНУ 8.153 ДР		
					Перелік елементів	Літ.	Мас	Мас-штаб
Изм	Лист	№ докум	Подпись	Дата				
Розроб.		Жарков Є. В.						
Перевір.		Ніконова А. О.						
Т. Контр.		Ніконова А. О.				Лист		Листів
						ІННІ ЗНУ 8.1531з		
Н. Контр.		Верьовкін Л. Л						
Затверд.		Критська Т.В.						

ДОДАТОК Б

```
#define useDS18b20
#define useAHT10
#define HEATER_CTRL 1
----------
#define corr_T 0
#define corr_H 0
#define getTHperiod_DS 2000
#if (1==useDHT11 || useDHT22 == 1 )
#define getTHperiod 10000
#elif (1==useBME280)
#define getTHperiod 6000
#elif (1== useHTU21)
#define getTHperiod 8000
#elif (1== useAHT10)
#define getTHperiod 7000 #endif

#define T_CONTROL 0
#define WDT_ENABLE 0
//============================================================
#define LOAD_TIME 0
#define CLEAR_EEPROM 0
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 ds1307;
int rtc[7
//----------дисплей--------------------
#include <LCD_1602_RUS_ms.h>
LCD_1602_RUS lcd(0x27, 16, 2);
//---------таймер----------//
uint32_t CycleFanInterval =10800000 ;
uint32_t CycleFanDuration = 3600000 ;
#include "GyverTimer.h"
GTimer Tick1_Timer(MS, 300);
GTimer cycFanOnTimer(MS, CycleFanInterval);
GTimer cycFanOffTimer(MS, CycleFanDuration);
//---------- датчик 18b20----------
#if (useDS18b20 == 1 || useDS18b20 == 3 )
#include <microDS18B20.h>
#define DS_PIN 10 // пин датчика ds18b20 //sh!!
#define DS_CHECK_CRC [true] // true/false - проверка контрольной суммы принятых данных - надежнее, но тратит немного больше flash
MicroDS18B20 <DS_PIN> DSsens; // датчик подключен к выводу //m
unsigned long lastTHgetDS = 0; // счетчик
#endif
//--------------------AHT10----------4.2
#if (useAHT10 == 1)
#include <AHT10.h>
//AHT10 AHT10sens(AHT10_ADDRESS_0X39);
AHT10 AHT10sens(AHT10_ADDRESS_0X38);
#endif
//---------------------------
unsigned long lastTHget = 0;
///----------
//--------watchdog-----
#if (1==WDT_ENABLE)
#include <avr/wdt.h>
#endif
//-------------------------------
//#define ground_Dpin 13// контакт датчика вологості почвы
#define ground_Apin A1
int GsensDRY = 600;
int GsensWET = 300;
#define AskSensGroundTime 600
unsigned long LastAskGS = 0;
//--- кнопки----
#include <AmperkaKB_ms.h>

#include <EEPROMex.h>
#include <EEPROMVar.h>

boolean DAY = 0;
#define Relay_Lamp 12
```

```
#define Relay_Heat 11
#define Relay_Water 7
#define Relay_Vent 8
#define Relay_Hum 16
#define Light_sens 17

float Temperature = 0;
byte Humidity = 0;
float tempV = 0;
boolean heating = 1;
float Temp_Box_Day = 24.0;
float Temp_Box_Night = 24.0;
float delta_Temp = 0.6;
float step_Temp = 0.2;

boolean lighting = 1;
byte start_day_h = 6;
byte start_day_m = 00;
byte duration_day_h = 18 ;
byte duration_day_m = 00;
byte offL_h;
byte offL_m ;
boolean sens_lighting = 0;
boolean No_Sunny = 0;
#define AskSensLighTime
unsigned long LastAskLS = 0;
boolean watering = 0;
byte start_wat_h = 8;
byte start_wat_m = 00;
byte duration_wat_h = 1 ;
byte duration_wat_m = 00;
byte offW_h;
byte offW_m ;
boolean ventilation = 1;
float Temp_Vent_ON = 30.0;
float delta_Temp_Vent = 2.0;
#define EE_cycleVentMode 60
#define EE_CycleFanInterval 61
#define EE_CycleFanDuration 65
bool cycleVentMode = 0;
bool cycfanON = 0;
//-3.3----------
boolean sens_watering = 0;
                - eeprom= 40
byte ground_Hum ;
boolean vkl = 0;
boolean vklS = 0;
#define AskSensTime 60000//300000
unsigned long LastAskSens = 0;
unsigned long PolivTime = 20000;
unsigned long PolivOn = 0;
byte firstAskPoliv = 1;
byte HumGND_ON = 45;
#define CONFIRM_COUNT 3
byte confirm_poliv;

boolean humidification = 1
byte Hum_ON = 50;
byte delta_Hum = 10;
byte firstRunDevice = 1;
unsigned long startMenuTime;
#define exitTime 5000
#define NUM_SETTINGS_MODES 6

//String Temp , TempD, TempN;

#if (1==T_CONTROL) //4.2
void (*resetSystem) (void) = 0;
#endif
//---------------------------------
//****************************************
//*************ФУНКЦІЇ*********************
//****************************************
void displays() {
  byte disp = 1;
  byte firstrun = 1;
  String TempD = roundFloat(Temp_Box_Day, 4, 1);
```

```
  String  TempN = roundFloat(Temp_Box_Night,  4, 1);

  startMenuTime = millis();
  //  KB.read();
  //   KB.getNum=0;
  KB.read();

  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime  )
   {
    lcd.clear();
    break;
   }
     //----------
#if (1== WDT_ENABLE )
   wdt_reset();
#endif
   //----------
   KB.read();

   if ( (KB.justPressed() && KB.getNum == 4) || firstrun == 1) {
    startMenuTime = millis();

    disp++;
    if (firstrun  == 1)firstrun  = 0;
    if (disp > 9)disp = 1;
    lcd.clear();
   }

   switch (disp) {
    case 1:
     {
      String Temp = roundFloat(Temperature,  4, 1);
      String TempV = roundFloat(tempV,  4, 1); //sh !!
      lcd.setCursor(0,  0);
      lcd.print((String)  zero(rtc[2]) + ":" + zero(rtc[1])  + F(" ТЕМП=") + Temp + char(223)  );
      lcd.setCursor(0,  1);
      if (1 == HEATER_CTRL) { // если режим контроля обогревателя //sh
       lcd.print((String)F("ВП=  ") + Humidity + "%" + "  " + F("ТН= ") + TempV + char(223)); //
       // lcd.print((String)  Humidity  + "%" + "/" +  TempV + char(223)+  "/" + ground_Hum  + "% ); //!!
      }
      else {
       lcd.print((String)F("ВП=  ") + Humidity + "%" + "  " + F("ВП= ") + ground_Hum  + "%  "); //3.3
      }
      // lcd.print((String)F("ВОЛОГІСТЬ=  ") + Humidity + "%");
      break;
     }
    case 2:
     lcd.setCursor(0,  0);
     lcd.print(F("ОБЛАДНАННЯ"));
     lcd.setCursor(0,  1);
     lcd.print((String)"L"  + curR(Relay_Lamp) + " T" + curR(Relay_Heat)  + " W" + curR(Relay_Water)  + " V" + curR(Relay_Vent)  + " H" +
curR(Relay_Hum));
     break;

    case 3:
     lcd.setCursor(0,  0);
     lcd.print((String)  F("СВІТЛО: ON  ") +  zero(start_day_h)  + ":" + zero(start_day_m));
     lcd.setCursor(0,  1);
     lcd.print((String)zero(duration_day_h)   + ":" + zero(duration_day_m)  + F(" OFF ") + zero(offL_h) + ":" + zero(offL_m));
     break;
    case 4:
     lcd.setCursor(0,  0);
     lcd.print((String)  F("ТЕМП: ДЕНЬ  ") +  TempD);
     lcd.setCursor(0,  1);
     lcd.print((String)  F("d=") + delta_Temp + F(" НІЧ ") +  TempN );
     break;
    case 5:
     lcd.setCursor(0,  0);
     lcd.print((String)  F("ПОЛИВ: ON ") +  zero(start_wat_h)  + ":" + zero(start_wat_m));
     lcd.setCursor(0,  1);
     lcd.print((String)zero(duration_wat_h)   + ":" + zero(duration_wat_m)  + F(" OFF ") + zero(offW_h) + ":" + zero(offW_m));  //
     break;
    case 6:
     lcd.setCursor(0,  0);
     lcd.print(F("ВЕНТИЛЯЦІЯ"));
     lcd.setCursor(0,  1);
```

```
      lcd.print((String)   F("ТЕМП:") +  Temp_Vent_ON + F(" d=") + delta_Temp_Vent);
      break;
     case 7:
      lcd.setCursor(0,  0);
      lcd.print(F("ЦИКЛ.  ВЕНТ."));//
      if(cycleVentMode)lcd.print(F("-ON"));
      else lcd.print(F("-OFF"));
      lcd.setCursor(0,  1);
      lcd.print(F("И="));
      lcd.print((long)CycleFanInterval  / 1000 / 60 / 60);
      lcd.print(F(":"));
      lcd.print((long)(CycleFanInterval  / 1000 / 60) % 60);
      lcd.print(F("  П="));
      lcd.print((long)CycleFanDuration   / 1000 /  60 / 60);
      lcd.print(F(":"));
      lcd.print((long)(CycleFanDuration   / 1000 /  60) % 60);
      break;
*/
     case 7:
      lcd.setCursor(0,  0);
      lcd.print(F("ЦИКЛ.  ВЕНТ."));
      if(cycleVentMode)lcd.print(F("-ON"));
      else lcd.print(F("-OFF"));
      lcd.setCursor(0,  1);
      lcd.print(F("И="));
      lcd.print((long)CycleFanInterval  / 1000 / 60 / 60);
      lcd.print(F("  П="));
      lcd.print((long)CycleFanDuration   / 1000 /  60 / 60);
      break;
     case 8:
      lcd.setCursor(0,  0);
      lcd.print(F("ЗВОЛОЖЕННЯ ПОВ."));
      lcd.setCursor(0,  1);
      lcd.print((String)   F("ВОЛОГ:") +  Hum_ON + F("% d=") + delta_Hum);
      break;
     case 9:
      lcd.setCursor(0,  0); lcd.print(F("L  LD T WT WD V H"));   //3.4
      lcd.setCursor(0,  1); lcd.print(curSet(lighting));
      lcd.setCursor(2,  1); lcd.print(curSet(sens_lighting));    //3.4
      lcd.setCursor(5,  1); lcd.print(curSet(heating));
      lcd.setCursor(7,  1); lcd.print(curSet(watering));
      lcd.setCursor(10,  1); lcd.print(curSet(sens_watering));
      lcd.setCursor(13,  1); lcd.print(curSet(ventilation));
      lcd.setCursor(15,  1); lcd.print(curSet(humidification));

      break;
    }

    delay(30);
  } //while
  KB.getNum = 0; // обнуляем номер клавиши
}

//--------------------------------------
String zero(int  data) {
  String str;
  if (data < 10) str = "0" + (String)data;
  else str = (String)data;
  //Serial.println((String)"str="+str);  // выводим
  return  str;
}
// Зчитуваня стану
char curR(int  pin) {
  if (digitalRead(pin)  == LOW) return  '+';
  else if  (digitalRead(pin)  == HIGH)   return  '-';
  else  return  'E';
}

String curSet(byte per) {
  if (per == 1) return  "+";
  else if  (per == 0)   return  "-";
  else  return  "err";
}

//---------------------------------------------------
// Робота з EEPROM
void update_EEPROM(int cod)
```

```
{
 switch (cod) {
  case 0:
   EEPROM.write(31, heating);
   EEPROM.writeFloat(1, Temp_Box_Day);
   EEPROM.writeFloat(5, Temp_Box_Night);
   EEPROM.writeFloat(9, delta_Temp);

   EEPROM.write(32, lighting);
   EEPROM.write(13, start_day_h);
   EEPROM.write(14, start_day_m);
   EEPROM.write(15, duration_day_h);
   EEPROM.write(16, duration_day_m);

   EEPROM.write(33, watering);
   EEPROM.write(17, start_wat_h);
   EEPROM.write(18, start_wat_m);
   EEPROM.write(19, duration_wat_h);
   EEPROM.write(20, duration_wat_m);
   //3.2
   EEPROM.write(70, ventilation);
   EEPROM.writeFloat(71, Temp_Vent_ON);
   EEPROM.writeFloat(75, delta_Temp_Vent);
   //3.3
   EEPROM.write(40, sens_watering);
   EEPROM.write(41, humidification);
   EEPROM.write(42, Hum_ON);
   EEPROM.write(43, delta_Hum);
   EEPROM.write(44, sens_lighting);
   EEPROM.write(45, HumGND_ON);
   EEPROM.writeInt(46, GsensDRY);
   EEPROM.writeInt(48, GsensWET);
   EEPROM.writeLong(50, PolivTime);

   EEPROM.writeLong(EE_cycleVentMode, cycleVentMode);
   EEPROM.writeLong(EE_CycleFanInterval, CycleFanInterval);
   EEPROM.writeLong(EE_CycleFanDuration,  CycleFanDuration);

   break;

  case 1:  EEPROM.updateFloat(cod, Temp_Box_Day); break;
  case 5:  EEPROM.updateFloat(cod, Temp_Box_Night);  break;
  case 9:  EEPROM.updateFloat(cod, delta_Temp); break;
  case 13:  EEPROM.updateByte(cod, start_day_h); break;
  case 14:  EEPROM.updateByte(cod, start_day_m); break;
  case 15:  EEPROM.updateByte(cod, duration_day_h);  break;
  case 16:  EEPROM.updateByte(cod, duration_day_m);  break;
  case 17:  EEPROM.updateByte(cod, start_wat_h);  break;
  case 18:  EEPROM.updateByte(cod, start_wat_m); break;
  case 19:  EEPROM.updateByte(cod, duration_wat_h);  break;
  case 20:  EEPROM.updateByte(cod, duration_wat_m);  break;
  case 31:  EEPROM.updateByte(cod, heating); break;
  case 32:  EEPROM.updateByte(cod, lighting); break;
  case 33:  EEPROM.updateByte(cod, watering); break;
  case 70:  EEPROM.updateByte(cod, ventilation);  break;
  case 71:  EEPROM.updateFloat(cod, Temp_Vent_ON); break;
  case 75:  EEPROM.updateFloat(cod, delta_Temp_Vent); break;
  case 40:  EEPROM.updateByte(cod, sens_watering); break;
  case 41:  EEPROM.updateByte(cod, humidification);  break;
  case 42:  EEPROM.updateByte(cod, Hum_ON); break;
  case 43:  EEPROM.updateByte(cod, delta_Hum); break;
  case 44:  EEPROM.updateByte(cod, sens_lighting); break;
  case 45:  EEPROM.updateByte(cod, HumGND_ON);  break;
  case 46:  EEPROM.updateInt(cod, GsensDRY); break;
  case 48:  EEPROM.updateInt(cod, GsensWET); break;
  case 50:  EEPROM.updateLong(cod, PolivTime); break;

  case EE_cycleVentMode: EEPROM.updateLong(cod, cycleVentMode); break;
  case EE_CycleFanInterval: EEPROM.updateLong(cod, CycleFanInterval); break;
  case EE_CycleFanDuration:  EEPROM.updateLong(cod, CycleFanDuration); break;
 }
}
//**************************************************************************************************
//********************************** SETUP ***********************************************
//**************************************************************************************************
void setup() {
```

```
  // Serial.begin(9600);
  pinMode(Relay_Lamp, OUTPUT);
  pinMode(Relay_Heat, OUTPUT);
  pinMode(Relay_Water, OUTPUT);
  pinMode(Relay_Vent, OUTPUT);//3.2
  pinMode(Relay_Hum, OUTPUT);//3.3

  // pinMode(ground_Dpin, INPUT);
  pinMode(ground_Apin, INPUT);
  analogReference(DEFAULT);

  digitalWrite(Relay_Lamp, HIGH);
  digitalWrite(Relay_Heat, HIGH);
  digitalWrite(Relay_Water, LOW);
  digitalWrite(Relay_Vent, LOW);
  digitalWrite(Relay_Hum, LOW);

  //=========таймер==========
  cycFanOnTimer.stop();
  cycFanOffTimer.stop();
  //==============================
  //-------Setup DS1307 RTC----------------------
#ifdef AVR
  Wire.begin();
#else
  Wire1.begin(); // Shield I2C pins connect to alt I2C bus on Arduino
#endif

  if (! ds1307.begin()) {
     // Serial.println(""Couldn't find RTC"");
  }

  if (! ds1307.isrunning()) {
   // Serial.println("RTC is NOT running!");

  }
  if (LOAD_TIME == 1) ds1307.adjust(DateTime(__DATE__, __TIME__));
  load();

  KB.begin(KB1x4);
  //-----------------------------------------

  No_Sunny = digitalRead(Light_sens);
  //============================================

#if (useDHT11 == 1 || useDHT22 == 1 )
  DHTsens.begin();
#endif
  //--------------------
#if (1==useBME280) Wire.begin();
  if (!BMEsens.begin())
  {
   lcd.clear();
   lcd.setCursor(0, 0);

     } else {

   BMEsens.resetToDefaults();
   BMEsens.writeOversamplingTemperature(BMx280MI::OSRS_T_x16);
   BMEsens.writeOversamplingHumidity(BMx280MI::OSRS_H_x16);//if sensor is a BME280, set an oversampling setting for humidity measurements.
  }
#endif
  //-------------------------------------------------------

#if (1==useHTU21)
  if (HTUsens.begin() == false)
  {
   lcd.clear(); //чистим экран
   lcd.setCursor(0, 0);
   lcd.print(F("HTU21 ERROR") );

   // Serial.println(F("HTU21D, SHT21 sensor is faild or not connected")); //(F()) saves string to flash & keeps dynamic memory free
  } else HTUsens.readTemperature();
#endif

#if (1==useAHT10)
  if (AHT10sens.begin() != true) {
```

```
   lcd.clear(); //чистим экран
   lcd.setCursor(0,  0);
   lcd.print(F("AHT10  ERROR") );

  }
#endif
  //=====================================================

  //--------watchdog-----4.2
#if (1==WDT_ENABLE )
  wdt_disable();
  delay(4000);  wdt_enable (WDTO_8S);

#endif

  //=======================EEPROM=======================

  if (1 == CLEAR_EEPROM) EEPROM.writeByte(100,  0); //

  if (EEPROM.read(100) != 1) {
   EEPROM.writeByte(100,  1);
   update_EEPROM(0);
  }
  else {
   heating = EEPROM.read(31);
   Temp_Box_Day = EEPROM.readFloat(1);
   Temp_Box_Night  = EEPROM.readFloat(5);
   delta_Temp = EEPROM.readFloat(9);

   lighting = EEPROM.read(32);
   start_day_h = EEPROM.read(13);
   start_day_m = EEPROM.read(14);
   duration_day_h  = EEPROM.read(15);
   duration_day_m  = EEPROM.read(16);

   watering = EEPROM.read(33);
   start_wat_h  = EEPROM.read(17);
   start_wat_m = EEPROM.read(18);
   duration_wat_h  = EEPROM.read(19);
   duration_wat_m  = EEPROM.read(20);

   ventilation  = EEPROM.read(70);
   Temp_Vent_ON = EEPROM.readFloat(71);
   delta_Temp_Vent = EEPROM.readFloat(75);

   sens_watering = EEPROM.read(40);
   humidification  = EEPROM.read(41);
   Hum_ON = EEPROM.read(42);
   delta_Hum = EEPROM.read(43);
   sens_lighting = EEPROM.read(44);
   HumGND_ON = EEPROM.read(45);
   GsensDRY = EEPROM.readInt(46);
   GsensWET = EEPROM.readInt(48);
   PolivTime = EEPROM.readLong(50);

   cycleVentMode = EEPROM.read( EE_cycleVentMode); //sh
   CycleFanInterval = EEPROM.readLong( EE_CycleFanInterval);
   CycleFanDuration  = EEPROM.readLong( EE_CycleFanDuration);

  }
  //=========таймер (EEPROM)==========
  cycFanOnTimer.setInterval(CycleFanInterval);
  cycFanOffTimer.setInterval(CycleFanDuration);
  cycFanOnTimer.stop();
  cycFanOffTimer.stop();
  //================================

}//setup
  //*************************************************************************************************
//***************************** LOOP ************************************************
//*************************************************************************************************

void loop() {

   //-----------------------------------------
#if (1== WDT_ENABLE )
  wdt_reset();       // watchdog
```

```
#endif
  //-----------------------------------------------
  delay(20);
  KB.read();
  if (KB.getNum == 1 )setup_menu();
  if (KB.getNum == 4 )displays();

  //================================================================================
  day_night();
  if (millis()  - LastAskLS > AskSensLighTime) {
   LastAskLS = millis();
   No_Sunny = digitalRead(Light_sens);
  }
  //================================================================================
  if (firstRunDevice)  {

    if (cycleVentMode) {
      cycfanON = 1;
      cycFanOnTimer.stop();
      cycFanOffTimer.start();
    }
    //-----------
  }
  //====== test =========
 if (0 == CycleFanInterval) {
  bool stat=cycFanOnTimer.isEnabled();
   CycleFanInterval = 20000;
   cycFanOnTimer.setInterval(CycleFanInterval);
   if(!stat)cycFanOnTimer.stop();
  }
  if (0 == CycleFanDuration)  {
   bool stat=cycFanOffTimer.isEnabled();
   CycleFanDuration  = 20000;
   cycFanOffTimer.setInterval(CycleFanDuration);
   if(!stat)cycFanOffTimer.stop();//остановить  если не работал, т.к. setInterval()
  }

  //================ ====================================================

#if (useDS18b20 == 1 || useDS18b20 == 3 )
  if (millis()  - lastTHgetDS > getTHperiod_DS   || 1 == firstRunDevice) {
    lastTHgetDS = millis(); // сброс

    if (1 == firstRunDevice)  {
      DSsens.requestTemp(); delay (1000);
      if (1 == useDS18b20) { // DS18b20
        firstRunDevice  = 0;
      }
    }
    DSsens.requestTemp();
    Temperature  = DSsens.getTemp() + corr_T;
     }
#endif

  //------------ DS18b20 -------
#if (1==useDHT11 || 1==useDHT22 || 1==useBME280 || 1==useHTU21 || 1==useAHT10)

  if (millis()  - lastTHget > getTHperiod   || 1 == firstRunDevice)  {
    lastTHget = millis(); // запоминаем
    firstRunDevice  = 0;

    //-------------------------------------------------------------
#if (useDHT11 == 1 || useDHT22 == 1 )
    Humidity = (byte)DHTsens.readHumidity()  + corr_H;
    float TempDHT = DHTsens.readTemperature();
    if (1 == HEATER_CTRL) {
      tempV =  TempDHT ;
    }
    else {
      if (useDS18b20 != 3) Temperature = TempDHT + corr_T;  "
    }
#endif
    //-------------------------------------------------------------
#if (1==useBME280)

    if (BMEsens.measure())
    {
```

```
    for (byte i = 0; i < 10; i++) {
     if (BMEsens.hasValue()) {
       Humidity = BMEsens.getHumidity() + corr_H;
       float TempBME = BMEsens.getTemperature();
       if (1 == HEATER_CTRL) { // если режим контроля обогревателя
        tempV = TempBME ;
       }
       else {

        if (useDS18b20 != 3) Temperature = TempBME + corr_T;
       }

       break;
     }
     delay(100);
    }//for

  } else {

    // Serial.println("could not start measurement BME280, is a measurement already running?");
  }
#endif
  //------------------------------------------------------------
#if (1==useHTU21)
  Humidity = HTUsens.readCompensatedHumidity() + corr_H;
  float TempHTU = HTUsens.readTemperature();
  if (1 == HEATER_CTRL) { // если режим контроля обогревателя //sh
   tempV = TempHTU ;
  }
  else {
   if (useDS18b20 != 3) Temperature = TempHTU + corr_T;
  }

  //  if (useDS18b20 != 3) Temperature = TempHTU + corr_T;

#endif
  //------------------------------------------------------------

#if (1==useAHT10)
  float TempAHT10 = AHT10sens.readTemperature(AHT10_FORCE_READ_DATA
  Humidity = AHT10sens.readHumidity(AHT10_USE_READ_DATA) + corr_H;
  if (1 == HEATER_CTRL) {
   tempV = TempAHT10 ;
  }
  else {
   if (useDS18b20 != 3) Temperature = TempAHT10 + corr_T;
  }

#endif

 } // if таймер

#endif
 //-------------------------------------
#if (1==T_CONTROL) //4.2
 if (Temperature > 150.0 || Temperature < -50.0 || isnan(Temperature)) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("TEMP ERROR") );
  lcd.setCursor(0, 1);
  lcd.print(F("reset...") );

  delay(2000);
  resetSystem();
 }
#endif
 //---------------------------------------------

 //========= end ======= ====================================================

 // ====================

 if (millis() - LastAskGS > AskSensGroundTime) {
  LastAskGS = millis();

  int G_read = analog_aver(ground_Apin, 10, 5);
  if (G_read < GsensWET) G_read = GsensWET;
```

```
 else if (G_read > GsensDRY)  G_read = GsensDRY;
 ground_Hum  = map(G_read, GsensWET, GsensDRY, 100, 0);
}
//*********************************************************
//***********************************
//*********************************************************
if (Tick1_Timer.isReady()) {
 String Temp = roundFloat(Temperature,  4, 1);
 // String  TempD = roundFloat(Temp_Box_Day,  4, 1);
 //String  TempN = roundFloat(Temp_Box_Night,  4, 1);
 String TempV = roundFloat(tempV,  4, 1); //sh !!
 // lcd.clear();
 // lcd.setCursor(0,  0);
 // lcd.print((String)  zero(rtc[2])  + ":" + zero(rtc[1])  + F(" ТЕМП=") + Temp + char(223) );
 lcd.setCursor(0,  0); lcd.print((String)  zero(rtc[2])  + ":" + zero(rtc[1]));
 lcd.setCursor(5,  0); lcd.print((String)  F(" ТЕМП=") + Temp +  char(223) );
 lcd.setCursor(0,  1);
 if (1 == HEATER_CTRL) { // если режим контроля обогревателя //sh
  lcd.print((String)F("ВП=  ") + Humidity  + "%" + " " + F("ТП= ") + TempV + char(223)); //
  // lcd.print((String)  Humidity + "%" + "/" +  TempV + char(223)+ "/" + ground_Hum  + "% "); //!!
 }
 else {
  lcd.print((String)F("ВП=  ") + Humidity  + "%" + " " + F("ВП= ") + ground_Hum  + "%  "); //3.3
 }
 //lcd.print((String)F("ВОЛОГІСТЬ=  ") + Humidity + "%");

 //*********************************************************
 //***************************************
 //*********************************************************
 //=======================================================
 if (watering == 1) {
  poliv();
 }
 else if (sens_watering == 0) digitalWrite(Relay_Water,  LOW);
 if (sens_watering == 1 && vkl == 0) {
  if (millis()  - LastAskSens > AskSensTime || 1 == firstAskPoliv) {
   LastAskSens = millis();
   firstAskPoliv = 0;
   poliv_Sen();
  }
  else if (millis() - PolivOn > PolivTime) {
   digitalWrite(Relay_Water,  LOW);
   //  Serial.println("pump  OFF!!!!!!!!!!!!!!!!!!!");
  }
 }
 else if (vkl == 0) {
  digitalWrite(Relay_Water,  LOW);
 }
 //=====================================================================
 //================================

 if (lighting == 1) {
  if (sens_lighting  == 1) {
   if (DAY == 1 && No_Sunny == 1 ) {
    digitalWrite(Relay_Lamp,  HIGH);
   }
   else  {
    digitalWrite(Relay_Lamp,  LOW);
   }
  }
  else {
   if (DAY == 1 ) {
    digitalWrite(Relay_Lamp,  HIGH);
   }
   else  {
    digitalWrite(Relay_Lamp,  LOW);
   }
  }
 }
 else digitalWrite(Relay_Lamp,  LOW);
 //=========================================================================
 //==================== ========================================
 if (heating == 1) {//если активно
  if ( DAY == 1 && Temperature  < (Temp_Box_Day  - delta_Temp)) {
   digitalWrite(Relay_Heat,  HIGH);
  }
  else if ( DAY == 0 && Temperature < (Temp_Box_Night  - delta_Temp)) {
```

```
      digitalWrite(Relay_Heat,  HIGH);
    }

    else if (DAY == 1 && Temperature > (Temp_Box_Day)) { // если день и подогрели
      digitalWrite(Relay_Heat,  LOW);
    }
    else if (DAY == 0 && Temperature > (Temp_Box_Night))  {
      digitalWrite(Relay_Heat,  LOW);
    }
   }
   else {
    digitalWrite(Relay_Heat,  LOW);
   }
   //=========================================================================

   //=========================================================
   if (ventilation  == 1) {

    if (cycleVentMode) {  //sh
     if (cycFanOnTimer.isReady()) {
       cycfanON = 1;
       cycFanOnTimer.stop();
       cycFanOffTimer.start();

     }
     if (cycFanOffTimer.isReady()) {
       cycfanON = 0;
       cycFanOffTimer.stop();
       cycFanOnTimer.start();

     }

    }

    float T = 0;
    if (1 == HEATER_CTRL)T = tempV;
    else T = Temperature;
    if (cycfanON || T > Temp_Vent_ON) {
     digitalWrite(Relay_Vent,  HIGH);
       }

    else if (T <= (Temp_Vent_ON - delta_Temp_Vent)) {
     digitalWrite(Relay_Vent,  LOW);
        // Serial.println((String)  Temp_Vent_ON + "-" + delta_Temp_Vent );
    }
   }
   else {
    digitalWrite(Relay_Vent,  HIGH);
   }
   //=========================================================================

   //=================================================
   if (humidification  == 1) {
    if ( Humidity  < Hum_ON) {
     digitalWrite(Relay_Hum,  HIGH);
    }

    else if (Humidity >= (Hum_ON + delta_Hum)) {
     digitalWrite(Relay_Hum,  LOW);
     // Serial.println((String)  Hum_ON + "-" + delta_Hum  );
    }
   }
   else {
    digitalWrite(Relay_Hum,  LOW);
   }
   //=========================================================================
  } //Tick1_Timer

  //*********************************************************

} //loop
void day_night () {
  get_time();
  int cur_h = rtc[2]  ;
  int cur_min = rtc[1];
  offL_h = start_day_h + duration_day_h;
  if (offL_h >= 24)offL_h  = offL_h - 24;
```

```
 offL_m = start_day_m + duration_day_m;
 if (offL_m >= 60) {
  offL_m = offL_m - 60;
  offL_h = offL_h + 1;
  if (offL_h >= 24)offL_h = offL_h  - 24;
 }
 DAY = 0;
 //--------------------------
 if (start_day_h > offL_h && (cur_h >= start_day_h || cur_h <= offL_h) ) {

  if (cur_h == start_day_h) {
    if (cur_min >= start_day_m ) DAY = 1;
  }
  if (cur_h > start_day_h) DAY = 1;
  if (cur_h < start_day_h && cur_h <= offL_h) DAY = 1;

  if (cur_h == offL_h) {
    if (cur_min >= offL_m )  DAY = 0;
  }
  if (cur_h > offL_h && cur_h < start_day_h) DAY = 0;
 }

 else if (start_day_h <= offL_h && cur_h >= start_day_h && cur_h <= offL_h) {
  if (cur_h == start_day_h) {
    if (cur_min >= start_day_m ) DAY = 1;
  }
  if (cur_h > start_day_h) DAY = 1;

  if (cur_h == offL_h) {
    if (cur_min >= offL_m )  DAY = 0;
  }
  if (cur_h > offL_h) DAY = 0;
 }
}//------------------------
//-------------------------------------------------------
//---- --------------------
void poliv() {
 get_time();


 int cur_h = rtc[2]  ;
 int cur_min = rtc[1];

 offW_h = start_wat_h + duration_wat_h;
 if (offW_h >= 24) offW_h = offW_h - 24;

 offW_m = start_wat_m  + duration_wat_m;
 if (offW_m >= 60) {
  offW_m = offW_m - 60;
  offW_h = offW_h + 1;
  if (offW_h >= 24) offW_h = offW_h - 24;
 }
 if (start_wat_h  > offW_h && (cur_h >= start_wat_h  || cur_h <= offW_h) )
  if (cur_h == start_wat_h) {
    if (cur_min >= start_wat_m ) vkl = 1;
  }
  if (cur_h > start_wat_h) vkl = 1;
  if (cur_h < start_wat_h  && cur_h <= offW_h) vkl = 1;

  if (cur_h == offW_h) {
    if (cur_min >= offW_m ) vkl = 0;
  }
  if (cur_h > offW_h && cur_h < start_wat_h) vkl = 0;
  //итог
  if (vkl == 1)digitalWrite(Relay_Water,  HIGH);
  else if (vklS = 0)digitalWrite(Relay_Water,  LOW );
 }
 else if (start_wat_h <= offW_h && cur_h >= start_wat_h && cur_h <= offW_h) {
  if (cur_h == start_wat_h) {
    if (cur_min >= start_wat_m ) vkl = 1;
  }
  if (cur_h > start_wat_h) vkl = 1;
  //выключение
  if (cur_h == offW_h) {
    if (cur_min >= offW_m ) vkl = 0;
  }
```

```
   if (cur_h > offW_h) vkl = 0;

      if (vkl == 1)digitalWrite(Relay_Water,  HIGH);
   else if (vklS = 0) digitalWrite(Relay_Water,  LOW );
  }

}//---- -poliv

//---- --------------------
void poliv_Sen() {

  if (ground_Hum  < HumGND_ON) {

   if (confirm_poliv  < CONFIRM_COUNT && digitalRead(Relay_Water)  == 1 ) confirm_poliv++;
   if (confirm_poliv  >= CONFIRM_COUNT && digitalRead(Relay_Water)  == 1 ) {      digitalWrite(Relay_Water,  HIGH);
     vklS = 1;
     PolivOn = millis();
     confirm_poliv  = 0;
        // Serial.println("pump  ON!!!!!!!!!!!!!!!!!!!!");
   }
  }

  else if (ground_Hum  >= HumGND_ON && vkl == 0) {
   digitalWrite(Relay_Water,  LOW);
   vklS = 0;
   //Serial.println("pump  OFF sens  !!!!!!!!!!!!!!!!!!!");
   //   Serial.println("poliv_Sen2");
  }

  //++++++++++
   if ( digitalRead(ground_Dpin)  == 1) {
     digitalWrite(Relay_Water,  LOW);
     PolivOn = millis();
   }

   else if (digitalRead(ground_Dpin)  == 0 && vkl == 0) {
     digitalWrite(Relay_Water,  HIGH);
     //   Serial.println("poliv_Sen2");
   }
  */
}//---- END-poliv_Sen

//-------------------------------------------------------------------------------------------------------------------
  int mass[n];
  for (int i = 0; i <= n - 1; i++)
  {
   mass[i] = analogRead(analogPin);
   // Serial.println((String)i+"="+  mass[i]);
   unsigned long timer = 0;
   while (1) {
     if (millis()  - timer >= delayTime)
       break;
   }
  }
  int Min = mass[0];
  for (byte i = 0; i <= n - 1; i++)
  {
   if (mass[i] < Min)  Min = mass[i];
  }
  //Serial.println((String)"Min="+  Min);
  int Max = mass[0];
  for (byte i = 0; i <= n - 1; i++)
  {
   if (mass[i] > Max)  Max = mass[i];
  }
  // Serial.println((String)"Max="+  Max);
  long  Sum = 0;
  for (byte i = 0; i <= n - 1; i++)
  {
   Sum = Sum + mass[i];
  }
  //Serial.println((String)"Sum="+  Sum);
  int  aver = (float)(Sum - Min - Max) / (n - 2);
  //Serial.println((String)"aver="+  aver);
  //delay(400);
  return  aver;
```

```
}
//**************************************
//--------------- float ----------------------------------//sh
String roundFloat(float  f, byte StrLength, byte numAfter) {
 // static char outstr[10];
 char outstr[10];
 dtostrf(f, StrLength, numAfter, outstr);
 return ((String)outstr);
}
//--------------------------------------
void setup_menu() {
 // Serial.println((String)"setup_menu  ");
 char* set_modes[] = {
  "ОСВІТЛЕННЯ", "ОПАЛЕННЯ", "ОРОШЕННЯ", "ВЕНТИЛЯЦІЯ", "ЗВОЛОЖЕННЯ ПОВ.", "ДАТА/ЧАС", "ВХІД..."
 }; //0,1,2,3,4,5,

 byte setting_mode = 0;
 byte next_setting_mode;
 byte firstrun = 1;

 startMenuTime = millis();
 KB.read();
 KB.getNum = 0;
 KB.read();
 while (KB.getNum != 2) {
  if ( (millis() - startMenuTime) > exitTime )
  {
   KB.getNum = 0;
   setting_mode = 6;
   break;
  }
  //----------
#if (1== WDT_ENABLE )
  wdt_reset();       // watchdog
#endif
  //----------
  KB.getNum = 0;
  KB.read();

  // Serial.println((String)"KB.getNum=  "+KB.getNum);

  if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
   // Serial.println("if");
   startMenuTime = millis();
   //  KB.read();
   //   KB.getNum=0;
   lcd.clear();
   // Serial.println((String)"KB.getNum=  "+KB.getNum);

   if (firstrun  == 0) {
    setting_mode++;
   }
   if (setting_mode > NUM_SETTINGS_MODES) {
    setting_mode = 0;
   }
   lcd.setCursor(0, 0);
   lcd.print(set_modes[setting_mode]  );

   next_setting_mode = setting_mode + 1;

   if (next_setting_mode  > NUM_SETTINGS_MODES) {
    next_setting_mode  = 0;
   }
   //  Serial.println((String)"setting_mode=  "+setting_mode);
   firstrun = 0;
  }
  delay(30);
 } //while
 //KB.getNum=0;
 delay(300);
 //--------------------------------------
 //pick the mode
 switch (setting_mode) {
  case 0:
   set_light();
   break;
  case 1:
```

```
    set_heat();
    break;
   case 2:
    set_water();
    break;
   case 3:
    set_ventilation();
    break;
   case 4:
    set_humidification();
    break;
   case 5:
    set_time();
    break;
   case 6:
    //exit menu
    break;
  }
}//end setup_menu

//************************************************
//************ ************************************
void on_off(byte types) { //.
 // Serial.println((String)"on_of  ");
 char* set_modes[] = {
  "ОТКЛ", "ВКЛ"
 }; //0,1
 byte num_modes = 1;
 byte setting_mode;
 byte next_setting_mode;
 byte firstrun  = 1;
 switch (types) {
  case 0: setting_mode  = lighting;  break;
  case 1: setting_mode  = heating;  break;
  case 2: setting_mode  = watering;  break;
  case 3: setting_mode  = ventilation;  break;
  case 31: setting_mode  = cycleVentMode;  break;
  case 4: setting_mode  = sens_watering;  break;
  case 5: setting_mode  = humidification;  break;.
  case 6: setting_mode  = sens_lighting;  break;
 }


 //   Serial.println((String)"setting_mode  1-"+setting_mode);

 KB.read();
 startMenuTime = millis();
 KB.getNum = 0;
 KB.read();

 while (KB.getNum != 2) {
  if ( (millis() - startMenuTime) > exitTime  ) break;

  //----------
#if (1== WDT_ENABLE )
  wdt_reset();       // сброс  watchdog
#endif
  //----------
  KB.read();
  // Serial.println((String)"KB.getNum  1= "+KB.getNum);

  if ( (KB.justPressed()  && KB.getNum == 1)  || firstrun  == 1) {
   //  Serial.println("if");
   startMenuTime = millis();
   KB.read();
   lcd.clear();
   // Serial.println((String)"KB.getNum  2= "+KB.getNum);

   if (firstrun  == 0) {
    setting_mode++;
   }
   if (setting_mode  > num_modes) {
    setting_mode  = 0;
   }
   lcd.setCursor(0,  0);
   lcd.print(set_modes[setting_mode]);
```

```
    next_setting_mode = setting_mode + 1;

    if (next_setting_mode > num_modes) {
     next_setting_mode = 0;
    }
    // Serial.println((String)"setting_mode=  "+setting_mode);
    firstrun = 0;
   }
   delay(50);
  } //while
  if (setting_mode == 0) {
   switch (types) {
    case 0: lighting = 0;  break;
    case 1: heating = 0;  break;
    case 2: watering = 0;  break;
    case 3: ventilation = 0;  break;
    case 31: cycleVentMode = 0;  break;
    case 4: sens_watering = 0;  break;
    case 5: humidification = 0;  break;
    case 6: sens_lighting = 0;  break;
   }
  }
  else {
   switch (types) {
    case 0: lighting = 1;  break;
    case 1: heating = 1;  break;
    case 2: watering = 1;  break;
    case 3: ventilation = 1;  break;
    case 31: cycleVentMode = 1;  break;
    case 4: sens_watering = 1;  break;
    case 5: humidification = 1;  break;
    case 6: sens_lighting = 1;  break;
   }
  }
  lcd.clear();
  lcd.setCursor(0,  0);
  lcd.print((String)">>" + set_modes[setting_mode] + "<<");
  delay(1000);
}
//*****************************************************
void load() {
  //--дисплей 1602------
  lcd.init();                  // initialize  the lcd
  lcd.backlight();
  lcd.setCursor(3,  0);
  lcd.print(F("Load..."));
 #if (1== WDT_ENABLE )
  delay(3000);
#else
  delay(300);
#endif
}

//************************************
void test(byte types) {

  boolean cur_pos;
  boolean save_pos;
  byte firstrun = 1;

  switch (types) {
   case 0: save_pos = digitalRead(Relay_Lamp);    break;
   case 1: save_pos = digitalRead(Relay_Heat);    break;
   case 2: save_pos = digitalRead(Relay_Water);   break;
   case 3: save_pos = digitalRead(Relay_Vent);   break;
   case 4: save_pos = digitalRead(Relay_Hum);   break;
  }
  cur_pos = save_pos;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();

  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime ) break;
      //----------
#if (1== WDT_ENABLE )
```

```
   wdt_reset();       // сброс watchdog
#endif
   //----------
   KB.read();
   if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
     startMenuTime = millis();
     KB.read();
     if (firstrun  == 0) {
       cur_pos = !cur_pos;
     }

     if (cur_pos  == 0) {
       lcd.clear();
       lcd.setCursor(0,  0);
       lcd.print(F("ON"));
       switch  (types) {
         case 0: digitalWrite(Relay_Lamp,  cur_pos);  break;
         case 1: digitalWrite(Relay_Heat,  cur_pos); break;
         case 2: digitalWrite(Relay_Water,  cur_pos); break;
         case 3: digitalWrite(Relay_Vent,  cur_pos); break;
         case 4: digitalWrite(Relay_Hum,  cur_pos); break;
       }

     }
     else if (cur_pos == 1) {
       lcd.clear();
       lcd.setCursor(0,  0);
       lcd.print(F("OFF"));
       switch  (types) {
         case 0: digitalWrite(Relay_Lamp,  cur_pos);  break;
         case 1: digitalWrite(Relay_Heat,  cur_pos); break;
         case 2: digitalWrite(Relay_Water,  cur_pos); break;
         case 3: digitalWrite(Relay_Vent,  cur_pos); break;
         case 4: digitalWrite(Relay_Hum,  cur_pos); break;
       }
     }
     firstrun  = 0;
   }
   delay(50);
  }
  switch (types) {
   case 0: digitalWrite(Relay_Lamp,  save_pos);  break;
   case 1: digitalWrite(Relay_Heat,  save_pos); break;
   case 2: digitalWrite(Relay_Water,  save_pos); break;
   case 3: digitalWrite(Relay_Vent,  save_pos); break;
   case 4: digitalWrite(Relay_Hum,  save_pos); break;
  }
}
//*****************************************************
void set_heat() {
  //Serial.println((String)"set_light  ");
  char* set_modes[] = {
    "ON/OFF", "ТЕМПЕРАТУРА ДЕНЬ", "ТЕМПЕРАТУРА НІЧ", "ВІДХИЛЕННЯ", "ТЕСТ"
  }; //0,1,2,3,4
  byte num_modes = 4; //0-4
  byte setting_mode = 0;
  byte next_setting_mode;
  byte firstrun  = 1;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
    if ( (millis() - startMenuTime) > exitTime  )
    {
      setting_mode = 5;
      break;
    }
       //----------
#if (1== WDT_ENABLE )
   wdt_reset();       // сброс watchdog
#endif
   //----------
   KB.read();

   if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
     startMenuTime = millis();
```

```
    KB.read();
    lcd.clear();

    if (firstrun  == 0) {
     setting_mode++;
    }
    if (setting_mode  > num_modes) {
     setting_mode  = 0;
    }
        lcd.setCursor(0,  0);
    lcd.print(set_modes[setting_mode]);
    next_setting_mode  = setting_mode  + 1;
    if (next_setting_mode  > num_modes) {
     next_setting_mode  = 0;
    }
    //  Serial.println((String)"setting_mode=   "+setting_mode);
    firstrun  = 0;
   }
   delay(50);
  } //while
  KB.getNum = 0;
  delay(300);
  //-------------------------------------
  //pick the mode
  switch (setting_mode)  {
   case 0:
    on_off(1);
    update_EEPROM (31);
    set_heat();
    break;
   case 1:
    Temp_Box_Day = setTH(0, -10, 90, Temp_Box_Day);
    update_EEPROM (1);
    set_heat();
    break;
   case 2:
    Temp_Box_Night  = setTH(0, -10, 90, Temp_Box_Night);
    update_EEPROM (5);
    set_heat();
    break;
   case 3:
   delta_Temp = setTH(0, 0, 10, delta_Temp);
    update_EEPROM (9);
    set_heat();
    break;
   case 4:
    test(1);
    set_heat();
    break;
   case 5:
    //exit menu
    break;
  }

} // set_heat
//*****************************************************
float setTH( byte TH, int minT, int maxT, float current_value  ) {
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();

  while (KB.getNum != 2) {

   if ( (millis() - startMenuTime) > exitTime  ) break;
   //----------
#if (1== WDT_ENABLE )
   wdt_reset();       // watchdog
#endif
   //----------
   KB.read();
   if ( KB.justPressed()  ) {
    startMenuTime = millis();
    KB.read();
    //   Serial.println((String)"KB.getNum   2= "+KB.getNum);
    if ( KB.getNum  == 3) {
     if (current_value  > minT) {
```

```
        if (TH == 0)current_value = current_value - step_Temp;
        else if (TH == 1)current_value = current_value - 1;
       }
       else {
        current_value = maxT;
       }
      }
      if ( KB.getNum == 4) {
       if (current_value < maxT) {
        if (TH == 0)current_value = current_value + step_Temp;
        else if (TH == 1) current_value = current_value + 1;
       }
       else {
        current_value = minT;
       }
      }
    }
    lcd.clear();
    //выводим
    lcd.setCursor(0, 0);
    lcd.print((String)current_value);

    // Serial.println((String)"current_value=  "+current_value);
    delay(50);
    // KB.read();
  } //while
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print((String)">>" + current_value + "<<");
  delay(1000);
  return current_value;
}

//*****************************************************
void set_humidification() {
  //Serial.println((String)"set_light  ");
  char* set_modes[] = {
   "ON/OFF", "ВОЛОГІСТЬ ON.", "ВІДХИЛЕННЯ", "ТЕСТ"
  }; //0,1,2,3

  byte num_modes = 3; //0-3
  byte setting_mode = 0;
  byte next_setting_mode;
  byte firstrun = 1;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime )
   {
    setting_mode = 5;
    break; //выходим
   }
   //----------
#if (1== WDT_ENABLE )
   wdt_reset();       // сброс watchdog
#endif
   //----------
   KB.read();
   if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
    startMenuTime = millis();
    KB.read();
    lcd.clear();
    if (firstrun == 0) {
     setting_mode++;
    }
    if (setting_mode > num_modes) {
     setting_mode = 0;
    }

    //выводим
    lcd.setCursor(0, 0);
    lcd.print(set_modes[setting_mode]);
    next_setting_mode = setting_mode + 1;
    if (next_setting_mode > num_modes) {
     next_setting_mode = 0;
```

```
    }
    //  Serial.println((String)"setting_mode=  "+setting_mode);
    firstrun = 0;
   }
   delay(50);
  } //while
  KB.getNum = 0;
  delay(300);
  //--------------------------------------
  //pick the mode
  switch (setting_mode) {
   case 0:
    on_off(5);
    update_EEPROM (41);
    set_humidification();
    break;
   case 1:
    Hum_ON = setTH(1, 1, 98, Hum_ON); //
    update_EEPROM (42);
    set_humidification();
    break;
   case 2:
    delta_Hum = setTH(1, 0, 90, delta_Hum);
    update_EEPROM (43);
    set_humidification();
    break;
   case 3:
    test(4);
    set_humidification();
    break;
   case 5:
    //exit menu
    break;
  }

} // set_heat
//*****************************************************
void set_light() {
  // Serial.println((String)"set_light  ");
  char* set_modes[] = {
   "ON/OFF", "ЧАС ВКЛЮЧЕННЯ", "ТРИВАЛІСТЬ", "ПОХИБКА ДАТЧИКА", "ТЕСТ" //3.4
  }; //0,1,2,3,4,5,

  byte  num_modes = 4; //0-4
  byte setting_mode = 0;
  byte next_setting_mode;
  byte firstrun = 1;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime )
   {
    setting_mode = 5;
    break;
   }
     //----------
#if (1== WDT_ENABLE )
   wdt_reset();       // сброс watchdog
#endif
   //----------
   KB.read();
   if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
    //  Serial.println("if");
    startMenuTime = millis();
    //  KB.read();
    lcd.clear();
    //  Serial.println((String)"KB.getNum  2= "+KB.getNum);
    if (firstrun  == 0) {
     setting_mode++;
    }
    if (setting_mode > num_modes) {
     setting_mode = 0;
    }
    //выводим
    lcd.setCursor(0,  0);
```

```
    lcd.print(set_modes[setting_mode]);
    next_setting_mode = setting_mode + 1;
    if (next_setting_mode > num_modes) {
     next_setting_mode = 0;
    }
    // Serial.println((String)"setting_mode=  "+setting_mode);
    firstrun = 0;
   }
   delay(50);
  } //while
  KB.getNum = 0;
  delay(300);
  //--------------------------------------
  //pick the mode
  switch (setting_mode) {
   case 0:
    on_off(0);
    update_EEPROM (32);
    set_light();
    break;
   case 1:
    onTime(0);
    update_EEPROM (13);
    update_EEPROM (14);
    set_light();
    break;
   case 2:
    set_duration(0);
    update_EEPROM (15);// НАДЕЮСЬ ДО СЮДОГО НИКТО НЕ ДОЧИТАЕТ
    update_EEPROM (16);
    set_light();
    break;

   case 3:
    on_off(6);
    update_EEPROM (44);
    set_light();
    break;
   case 4:
    test(0);
    set_light();
    break;
   case 5:
    //exit menu
    break;
  }

} // set_light

//************************************************

void set_water() {
  // Serial.println((String)"set_light  ");
  char* set_modes[] = {
   "ПО РОСКЛАДУ", "ЧАС ВКЛЮЧЕНИЯ", "ТРИВАЛІСТЬ", "ПО ДАТЧИКУ",  "ВОЛОГІСТЬ ВКЛ.", "ТЕСТ",  "КАЛІБРУВАННЯ" ,  "ТРИВ.ПО ДАТЧИК"
  }; //0,1,2,3,4,5, 6,7
  byte num_modes = 7; //0-4 //3.5//4,1
  byte setting_mode = 0;
  byte next_setting_mode;
  byte firstrun = 1;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime )
   {
    setting_mode = 8; // выход//4,1
    break;
   }
    //----------
#if (1== WDT_ENABLE )
   wdt_reset();       // сброс watchdog
#endif
   //----------
   KB.read();
   // Serial.println((String)"KB.getNum  1= "+KB.getNum);
```

```
 if ( (KB.justPressed() && KB.getNum == 1) || firstrun == 1) {
  // Serial.println("if");
  startMenuTime = millis();
  KB.read();
  lcd.clear();
  // Serial.println((String)"KB.getNum  2= "+KB.getNum);

  if (firstrun  == 0) {
   setting_mode++;
  }
  if (setting_mode > num_modes) {
   setting_mode = 0;
  }
  //выводим
  lcd.setCursor(0,  0);
  lcd.print(set_modes[setting_mode]);
  next_setting_mode = setting_mode + 1;
  if (next_setting_mode  > num_modes) {
   next_setting_mode  = 0;
  }
  // Serial.println((String)"setting_mode=  "+setting_mode);
  firstrun = 0;
 }
 delay(50);
} //while
delay(300);
//--------------------------------------
switch (setting_mode) {
 case 0:
  on_off(2);
  update_EEPROM (33);
  set_water();
  break;
 case 1:
  onTime(2);
  vkl = 0;
  update_EEPROM (17);
  update_EEPROM (18);
  set_water();
  break;
 case 2:
  set_duration(2);
  vkl = 0; // сброс полива 4.1
  update_EEPROM (19);
  update_EEPROM (20);
  set_water();
  break;
 case 3:
  on_off(4);
  update_EEPROM (40);
  set_water();
  break;
 case 4:
  HumGND_ON = setTH(1, 1, 100, HumGND_ON);
  update_EEPROM (45);
  set_water();
  break;
 case 5:
  test(2);
  set_water();
  break;
 case 6:
  GSkalibr();
  update_EEPROM (46);
  update_EEPROM (48);
  set_water();
  break;
 case 7:
  set_duration(3);
  update_EEPROM (50);
  set_water();
  break;
 case 8:
  //exit menu
  break;
}
```

```
} // set_light
//*************************************************
//*************************************************
void GSkalibr() {
  startMenuTime = millis();
  KB.read();
  //----------------
  lcd.clear();
  lcd.setCursor(0,  0);
  lcd.print("СУХОЙ"  );
  delay(2000);
  KB.getNum = 0;
  //----------------
  while (KB.getNum != 2) {
    if ( (millis() - startMenuTime) > 20000 ) {
      break; //выходим
    }
    //----------
#if (1== WDT_ENABLE )
    wdt_reset();       // сброс  watchdog
#endif
    //----------
    int val = analog_aver(ground_Apin,  10, 5);
    lcd.clear();
    lcd.setCursor(0,  0);
    lcd.print(val);
    delay(10);
    KB.read();

    if ( KB.justPressed() && KB.getNum == 2) {
      GsensDRY = val;
      lcd.clear();
      lcd.setCursor(0,  0);
      lcd.print((String)">>"  + GsensDRY + "<<");
      delay(2000);
      break;
    }
  }
  //*************************************
  startMenuTime = millis();
  KB.read();
  //----------------
  lcd.clear();
  lcd.setCursor(0,  0);
  lcd.print("В ВОДЕ" );
  delay(1000);
  KB.getNum = 0;
  //----------------
  while (KB.getNum != 2) {
    if ( (millis() - startMenuTime) > 20000 ) {
      break;
    }
    int val = analog_aver(ground_Apin,  10, 5);
    lcd.clear();
    lcd.setCursor(0,  0);
    lcd.print(val);
    delay(10);
    KB.read();
    if ( KB.justPressed() && KB.getNum == 2) {
      GsensWET = val;
      lcd.clear();
      lcd.setCursor(0,  0);
      lcd.print((String)">>"  + GsensWET + "<<");
      delay(1000);
      break;
    }
  }
}
//*************************************************************************
//set time and date routine / настройка часов
void set_time() {

  //fill settings with current clock values read from clock
  get_time();
  byte set_min   = rtc[1];
  byte set_hr    = rtc[2];
  byte set_date  = rtc[4];
```

```
  byte set_mnth  = rtc[5];
  int  set_yr    = rtc[6];

  //Set function - we pass in: which 'set' message to show at top, current value, reset value, and rollover limit.
  set_hr   = set_value(1, set_hr, 0, 23);
  set_min  = set_value(0, set_min, 0, 59);
  set_date = set_value(2, set_date, 1, 31);
  set_mnth = set_value(3, set_mnth, 1, 12);
  set_yr   = set_value(4, set_yr, 2015, 2099);
  ds1307.adjust(DateTime(set_yr, set_mnth, set_date, set_hr, set_min));
}
//************************************************************************
void get_time(){
  //get time
  DateTime now = ds1307.now();
  //save time to array
  rtc[6] = now.year();
  rtc[5] = now.month();
  rtc[4] = now.day();
  rtc[3] = now.dayOfTheWeek(); //3.5
  rtc[2] = now.hour();
  rtc[1] = now.minute();
  rtc[0] = now.second();
}
//************************************************************************

//used to set min, hr, date, month, year values. pass
//message = which 'set' message to print,
//current value = current value of property we are setting
//reset_value = what to reset value to if to rolls over. E.g. mins roll from 60 to 0, months from 12 to 1
//rollover limit = when value rolls over
int set_value(byte message, int current_value, int reset_value, int rollover_limit) {
  KB.read();
  char* messages[] = {
    "МИНУТЫ", "ЧАСЫ", "ДЕНЬ", "МЕСЯЦ", "ГОД", "СЕКУНДЫ"
  };

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(messages[message]);
  delay(1200);
  //Serial.println((String)"KB.getNum  0= "+KB.getNum);
  //Serial.println((String)"mes-  "+message);
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
    if ( (millis() - startMenuTime) > exitTime ) break;
    //----------
#if (1== WDT_ENABLE )
    wdt_reset();       // сброс watchdog
#endif
    //----------
    KB.read();
    // Serial.println((String)"KB.getNum  11= "+KB.getNum);

    if ( KB.justPressed() ) {
      startMenuTime = millis();
      KB.read();
      //    Serial.println((String)"KB.getNum  2= "+KB.getNum);
      if ( KB.getNum == 3) {
        if (current_value > reset_value) {
          current_value--;
        }
        else {
          current_value = rollover_limit;
        }
      }
      if ( KB.getNum == 4) {
        if (current_value < rollover_limit) {
          current_value++;
        }
        else {
          current_value = reset_value;
        }
      }
    }
```

```
   lcd.clear();
   //выводим
   lcd.setCursor(0,  0);
   lcd.print((String)current_value);

   //  Serial.println((String)"current_value=   "+current_value);
   delay(50);
   //  KB.read();
  } //while
  lcd.clear();
  lcd.setCursor(0,  0);
  lcd.print((String)">>"  + current_value  + "<<");
  delay(1000);
  return current_value;
}
//*************************************************************************
//**************************************************
void onTime(byte type) {
  byte set_hr;
  byte set_min;)
  // byte set_sec;
  switch (type) {
    case 0: set_hr = start_day_h;
      set_min = start_day_m;
      start_day_h   = set_value(1, set_hr, 0, 23);
      start_day_m  = set_value(0, set_min, 0, 59);
      break;
    case 2: set_hr = start_wat_h;
      set_min = start_wat_m;
      start_wat_h   = set_value(1, set_hr, 0, 23);
      start_wat_m  = set_value(0, set_min, 0, 59);
      break;
  }
}
//**************************************************

void set_duration(byte  Type) { //type:
  byte set_hr;
  byte set_min;

  switch (Type) {
    case 0: set_hr = duration_day_h; // свет
      set_min = duration_day_m;
      duration_day_h   = set_value(1, set_hr, 0, 23);
      duration_day_m   = set_value(0, set_min, 0, 59);
      break;
    case 2: set_hr = duration_wat_h;  // полив
      set_min = duration_wat_m;
      duration_wat_h    = set_value(1, set_hr, 0, 23);
      duration_wat_m   = set_value(0, set_min, 0, 59);
      break;
    case 3:
      {
        int mins = (long)PolivTime  / 1000 / 60;
        byte secs = (long)(PolivTime  / 1000) % 60;
        mins  = set_value(0, mins, 0, 240);
        secs  = set_value(5, secs, 0, 59);
        PolivTime = (long)mins  * 60 + secs;
        PolivTime = PolivTime * 1000;
      }
      break;

    *  case 41: // sh
      { int hrs = (long)CycleFanInterval / 1000 / 60 / 60;
        int mins = (long)(CycleFanInterval / 1000 / 60) % 60;
        hrs  = set_value(1, hrs, 0, 23);
        mins  = set_value(0, mins, 0, 59);
        CycleFanInterval = (long)hrs * 60 * 60 + (long)mins * 60; //  в сек
        CycleFanInterval = CycleFanInterval * 1000;
        cycFanOnTimer.setInterval(CycleFanInterval);
      }
      break;
      */
        case 41:
      {
        bool stat=cycFanOnTimer.isEnabled();
        // int hrs = (long)CycleFanInterval / 1000 ;
```

```
   int hrs = (long)CycleFanInterval / 1000 / 60 / 60;
      hrs  = set_value(1, hrs, 0, 23);
      if(0==hrs)CycleFanInterval=20000;
      else{
      //  CycleFanInterval = (long)hrs*1000;
      CycleFanInterval = (long)hrs * 60 * 60 ; // в сек
      CycleFanInterval = CycleFanInterval * 1000;
      }
         cycFanOnTimer.setInterval(CycleFanInterval);
    if(!stat)cycFanOnTimer.stop();
    }
    break;

   case 42:
    { int hrs = (long)CycleFanDuration  / 1000 / 60 / 60;
      int mins = (long)(CycleFanDuration  / 1000 / 60) % 60;
      hrs  = set_value(1, hrs, 0, 23);
      mins  = set_value(0, mins, 0, 59);
      CycleFanDuration = (long)hrs  * 60 * 60 + (long)mins  * 60;
      CycleFanDuration = CycleFanDuration  * 1000;
      cycFanOffTimer.setInterval(CycleFanDuration);
    }
    break;
    */
       case 42:
    {
      bool stat=cycFanOffTimer.isEnabled();
     //  int hrs = (long)CycleFanDuration  / 1000 ;
    int hrs = (long)CycleFanDuration / 1000 / 60 / 60;
       hrs  = set_value(1, hrs, 0, 23);
       if(0==hrs)CycleFanDuration=20000;
      else{
       // CycleFanDuration = (long)hrs*1000;
      CycleFanDuration = (long)hrs  * 60 * 60;
      CycleFanDuration = CycleFanDuration * 1000;
      }
      cycFanOffTimer.setInterval(CycleFanDuration);
      if(!stat)cycFanOffTimer.stop();
    }
    break;
  }
}
void set_ventilation() {
  //Serial.println((String)"set_light  ");
  char* set_modes[] = {
   "ON/OFF", "ТЕМПЕРАТУРА ON.", "ПОХИБКА", "ТЕСТ", "ЦИКЛІЧНО" , "ІНТЕРВАЛ ЦИКЛА", "ТРИВАЛ. ЦИКЛА"
  }; //0,1,2,3,4,5,6

  byte num_modes = 6; //0-6
  byte setting_mode = 0;
  byte next_setting_mode;
  byte firstrun  = 1;
  KB.read();
  startMenuTime = millis();
  KB.getNum = 0;
  KB.read();
  while (KB.getNum != 2) {
   if ( (millis() - startMenuTime) > exitTime  )
   {
     setting_mode = 7; // выход
     break; //выходим
   }
      //----------
#if (1== WDT_ENABLE )
    wdt_reset();       // сброс  watchdog
#endif
    //----------
    KB.read();
    if ( (KB.justPressed()  && KB.getNum == 1)  || firstrun  == 1) {
      startMenuTime = millis();
      KB.read();
      lcd.clear();
      if (firstrun  == 0) {
       setting_mode++;
      }
      if (setting_mode  > num_modes) {
       setting_mode  = 0;
```

```
    }
    //выводим
    lcd.setCursor(0,  0);
    lcd.print(set_modes[setting_mode]);
    if (setting_mode  == 4) {
     lcd.setCursor(3,  1);
     lcd.print("РЕЖИМ");
    }
    next_setting_mode  = setting_mode  + 1;
    if (next_setting_mode  > num_modes) {
     next_setting_mode  = 0;
    }
    //  Serial.println((String)"setting_mode=   "+setting_mode);
    firstrun  = 0;
   }
   delay(50);
  } //while
  KB.getNum = 0;
  delay(300);
  //-------------------------------------
  //pick the mode
  switch (setting_mode)  {
   case 0:
    on_off(3); //1- вент
    update_EEPROM (70);
    set_ventilation();
    break;
   case 1:
    Temp_Vent_ON = setTH(0,  0, 90, Temp_Vent_ON);
    update_EEPROM (71);
    set_ventilation();
    break;
   case 2:
    delta_Temp_Vent = setTH(0, 0, 50, delta_Temp_Vent);
    update_EEPROM (75);
    set_ventilation();
    break;
   case 3:
    test(3);
    set_ventilation();
    break;

   case 4: //sh
    on_off(31);
    update_EEPROM (EE_cycleVentMode);
    if (cycleVentMode) {
     cycfanON = 1;
     cycFanOnTimer.stop();
     cycFanOffTimer.start();
    }
    else {
     cycFanOffTimer.stop();
     cycFanOnTimer.stop();
     cycfanON = 0;
    }
    set_ventilation();
    break;
   case 5: //sh
    set_duration(41);
    update_EEPROM (EE_CycleFanInterval);
    set_ventilation();
    break;
   case 6: //sh
    set_duration(42);
    update_EEPROM (EE_CycleFanDuration);
    set_ventilation();
    break;
   case 7:
    //exit menu
    break;
  }
 } // set_ventilation
          //****************************************
```