ТЕЗИ

Калічак Юрій, 10 клас, Технічний ліцей м. Києва; педагогічний керівник: вчитель інформатики Стеценко Антоніна Іванівна.

Протягом останніх 10 років розвиток тепличних технологій різко популяризувався з огляду на свою перспективність і окупність. Вирощування овочів в закритому ґрунті актуальне, бо вони користуються великим попитом серед споживачів, особливо в зимовий період року. В сучасному світі потреби в екологічному та здоровому харчуванні зростають з кожним днем. Так з'явились гроубокси – кліматичні пристрої для вирощування зелені, салатів, ягід та овочів в домашніх умовах. Щоб полегшити цей процес, почали використовуватися різноманітні сучасні технології та принципи. Автоматизація будь-якого процесу при вирощуванні рослин в гроубоксі дозволяє знизити його залежність від людського фактора і збільшити врожайність за рахунок точності підтримки заданих параметрів.

При виникненні потреби в такому пристрої, у початківців фермерів виникає думка, що здійснювати процес управління гроубоксом своїми руками в домашніх умовах дуже складно, дорого і знадобиться для цього багато інструментів. Насправді, виготовлення такого пристрою не займе багато часу і коштів. Зібрати прилад можна потрібного розміру та з необхідними функціями. При складанні власними руками можна значно заощадити на дорогих матеріалах. Важливо пам'ятати і дотримуватися правил складання.

Тобто можливість автоматизувати процес вирощування в теплиці – серйозне полегшення для людини. В даному проекті розглядається створення універсального клімат-контролера для гроубокса, універсальність якого полягає в тому, що дану розробку можна використовувати для різних режимів кліматичного регулювання та спостереження.

Даний проект дає змогу швидко, просто і зручно контролювати процес вирощування рослин, зробити його повністю регульованим і максимально оптимальним.

Мета роботи: У результаті вивчення проблеми створити комплексний клімат-контролер для теплиці (гроубокса).

Актуальність роботи: Для багатьох господарників важливо контролювати процес вирощування рослин, зробити його повністю регульованим і максимально оптимальним. Цього можна досягти саме завдяки впровадженню системи комплексний клімат-контроль. Ці питання і розглядаються в даній роботі.

Об'єкт дослідження: Автоматичний контролер для теплиці (гроубокса) як сучасний пристрій, кліматичного регулювання та спостереження за рослинами, який є простим в управлінні і при цьому не вимагає додаткової участі людини.

Предмет дослідження: створення контролера для гроубокса, який є нескладним в користуванні та має досить простий і зрозумілий інтерфейс.

Результати: Досліджено систему гідропонного вирощування рослин, а також розроблено конструкцію гроубоксу. Розроблено програмне забезпечення для роботи системи керування процесом.

Створено програму: комплексний клімат-контролер для теплиці (гроубокса) зі зручним інтерфейсом, за допомогою якого можна просто і зручно контролювати процес вирощування рослин, зробити його повністю регульованим і максимально оптимальним.

ЗМІСТ

Вступ

Інтерес до вирощування свіжої зелені, овочів та фруктів в домашніх умовах зростає з кожним днем. Особливо це стало актуальним для тих, хто цікавиться здоровою та екологічно чистою їжею. Так з'явилися гроубокси – кліматичні пристрої для вирощування рослин. Гроубокс представляє собою повністю або частково закриту систему для вирощування розсади, зелені, овочів, квітів.

Лідерами тепличного бізнесу і першопрохідцями щодо впровадження інновацій можна вважати Нідерланди, які вивели його на якісно новий рівень. Завдяки власному дослідницькому центру, робота якого присвячена саме розробкам теплиць, голландці зуміли значно знизити викиди парникових газів і заощадити на енергоспоживанні. Тим самим голландські виробники вийшли на лідируючі позиції в світі з виробництва та експорту овочів закритого ґрунту. Не відстають у тепличному бізнесі США та Канада, вирощуючи найрізноманітніші продукти високої якості, походження яких можна з легкістю відстежити. Секрет успіху американців – використання передових технологій контролю за вирощуванням рослин, які не потребують особливого людського втручання. Причому американці і канадці в своєму виборі овочів не дуже різноманітні: приблизно до 50% ринку займає виробництво традиційних огірків і помідорів, для експорту – різні трави, салати та інші овочі першокласної якості. Цікавий приклад Німеччини, де більшість теплиць (43%, або 1,6 тис. га) побудовані у 1980-ті роки, але, незважаючи на це, були успішно модернізовані за останнім словом техніки. Слід зауважити, що більшість (48%) теплиць належать малому бізнесу і фермерам-одноосібникам (до 1000 кв. м), і тільки невелика частка (13%) – великим підприємствам (до 5000 кв. м). Більш того, попит на тепличні овочі в Німеччині останнім часом настільки виріс, що Федеральне міністерство сільського господарства навіть розробило стратегію розвитку рослинництва, пріоритетним напрямком якої став саме тепличний бізнес.

Однак українські реалії тепличного бізнесу не настільки позитивні, як, припустимо, в Європі. За даними Асоціації «Теплиці України», рентабельність такого бізнесу не перевищує 10%, а експортується приблизно 20% від усієї виробленої продукції. Згідно з офіційними даними, в 2018 році виробництво продукції закритого ґрунту в Україні склало 547 тис. тонн.

Найбільшим українським тепличним комплексом можна вважати Комбінат «Тепличний», що займає 48,5 га в Київській області. З 1994 року почалася його масштабна модернізація. Сьогодні в комплексі встановлено обладнання для зволоження повітря і крапельного поливу рослин з одночасною їх підгодівлею, обладнання для обігріву теплиць, налагоджено комп'ютерне регулювання мікроклімату. Впровадження передових технологій дозволяє досягати врожайності не гірше, ніж у голландців – до 61 кг з 1 кв. м. За рахунок тепличного комплексу ціни на овочі закритого ґрунту на внутрішньому ринку цілком доступні для вітчизняних споживачів.

Якщо впроваджувати тепличні технології правильно і поетапно, можна значно скоротити час дозрівання врожаю (до 30-35 днів), а також збільшити обсяги виробництва приблизно до 30%. Це копіткий і тривалий процес, однак результат говорить сам за себе.

Сучасні тепличні комплекси проектують з максимальною точністю, адже від правильно вибудуваного парникового ефекту будуть залежати температура всередині теплиці і кількість енергії, що витрачається. Усередині стін теплиці теплове випромінювання перетворюється в тепло, нагріваючи теплицю зсередини. Матеріали, з яких зроблена теплиця, також впливають на здатність утримувати тепло і регулювати температуру всередині теплиці. Однак є й альтернативні джерела тепла. Наприклад, поліетиленові матеріали здатні зберегти до 50% тепла, скло – усі 60%. Найчастіше клімат-система автоматизована, контролюється спеціальними термодатчиками, які при ознаках зниження температури здіснюють її коригування.

Висота теплиці також відіграє важливу роль у розвитку рослин, і інженери намагаються максимально збільшити кількість прохідного денного світла шляхом преломленої форми даху. Так сонячне світло буде з одного боку відбивати промені, з іншого – збирати енергію в світлові батареї, встановлені на даху. Своєю чергою, прогрітий за день ґрунт здатний утримувати тепло протягом ночі. У багатьох тепличних комплексах створюється власний мікроклімат, для підтриманя якого фермери використовують біологічні засоби захисту від шкідників. Це робить овочі органічними, без будь-яких домішок хімікатів і пестицидів. Всі необхідні мінеральні речовини подаються через спеціальні трубки до коренеплоду кожної рослини, все – в строго лімітованій кількості. Головна мета всіх хитрощів і технологій полягає в тому, щоб створити для рослин оптимальне середовище, яке не тільки скорочує час культивування, але й дозволить заощадити витратні фінанси.

1. Основна частина

1.1. Огляд готових конструкцій міні-теплиці

На сучасному ринку є великий вибір готових *міні-теплиць*, в яких можна вирощувати розсаду або зелень. Найпростіша модель – металеві стелажі, покриті чохлом з щільної поліетиленової плівки. Подібні конструкції представлені в широкому асортименті: є невеликі парники, які можна встановлювати на підвіконні, і більш місткі моделі, які можна поставити на балконі або прямо в житловому приміщенні біля вікна. З'явилися не тільки звичайні парники, але й автоматичні конструкції, які суттєво полегшують догляд за саджанцями.

До найпростішої конструкції відноситься шафа-теплиця, так званий *гроубокс*. Сам термін «гроубокс» утворений від англійського слова, що в перекладі означає ящик для вирощування рослин. Необхідний мікроклімат всередині забезпечується спеціальним чохлом на блискавці, а всередині такої шафи знаходиться кілька полиць для розміщення рослин. На жаль, розмір такої конструкції не дозволяє розміщувати її на підвіконні, але її цілком можна поставити на засклену балконі або лоджії.

Такі міні-теплиці можуть бути відкритими або закритими. Зовні вони нагадують вітрину або акваріум. Як правило, в такій ємності є не тільки датчики для підтримки температури, але й лампи освітлення, тому гроубокс можна ставити навіть в кімнаті, далеко від підвіконня.

Існують найсучасніші розробки – *аерогарде*, в якій процес вирощування рослин повністю автоматизований і проводиться без участі людини. Система обладнана системою вентиляції і спеціальними розпилювачами, які постачають корисні речовини безпосередньо до коренів рослин, тому культури в таких пристроях можна вирощувати навіть без ґрунту.

Подібні автоматичні системи коштують дорого, але при цьому дозволяють не тільки культивувати власну розсаду, а й цілий рік вирощувати свіжу зелень, овочі і квіти. Деякі з вищезазначених моделей призначені для вирощування рослин в субстраті, а інші працюють за принципом гідропонних установок.

Незалежно від обраного типу конструкції, існують певні вимоги, яким повинні відповідати домашні саморобні парники.

Щоб вирощування рослин було успішним, міні-теплиця повинна володіти такими характеристиками:

- забезпечувати стабільний і сприятливий мікроклімат для вирощуваних культур (температуру, вологість, освітлення і вентиляцію);
- конструкція повинна бути зручною для догляду за рослинами і збирання врожаю (якщо ви плануєте вирощувати в парнику зелень або овочі);
- володіти високою міцністю, оскільки зазвичай домашні парники використовуються протягом декількох сезонів.

Крім того, конструкція повинна гармонійно вписуватися в простір кімнати і не затінювати приміщення.

В таких моделях здійснюється автоматичний мониторинг температури і вологості. Датчики температури і вологості дозволяють здійснювати збір і відображеня даних з усієї площі теплиці. В автоматичному режимі відбувається провітрювання, вентиляція теплиці та зволоження.

Для цього використовуються вмонтовані автоматичні системи освітлення, вентиляції, фільтрації, підтримки температур і контролю вмісту CO_2. Закриті системи практично не потребують участі людини для підтримки оптимальних умов вирощування рослин.

Електронне управління полегшує роботу, а всі параметри виводяться на дисплей. Цифрові, світлові й графічні індикатори укупі зі звуковою

сигналізацією дозволяють відслідковувати процеси виведення в режимі реального часу.

1.2. Розробка проекту

Більшість проектів для створення контролера для гроубокса, які можна знайти в Інтернеті, мають певні недоліки. Аналогів таких регуляторів багато, але майже всі вони промислові, а тому досить дорогі та потребують наявності великої кількості додаткових пристроїв, що є незручним в домашніх умовах. В основі даної розробки знаходиться мікроконтролер. Його основна перевага – невеликий розмір. Окрім того, існуючі аналоги можуть містити помилки в програмному коді, а відтак можуть дати не очікуваний результат. Та й самі алгоритми досить громіздкі, і не завжди є оптимальними та результативними.

Тому виникла ідея створити свій проект для таких же цілей, але нескладний в користуванні з досить простим та зрозумілим інтерфейсом. Для створення програми було обрано мову програмування C++.

C++ – універсальна мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом у 1979 році та названа «Сі з класами». Страуструп перейменував мову у C++ у 1983 р. Базується на мові Сі. Визначена стандартом ISO/IEC 14882:2003.

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення.

Таким чином, робота над проектом проводилася за двома напрямками:

- створення електронного пристрою – контролера, що являється блоком, який відповідає за автоматизацію роботи гроубокса;
- написання програмного забезпечення для даного блоку.

Програмне забезпечення передбачає такі функціональні можливості :

1. ПІД (Пропорційний Інтегральний Диференційований) керування температурою.
2. Оптимальне використаня датчика вологості ґрунту (дешеві датчики швидко окислюються й приходять в непридатність).
3. Бережливе ставлення до вікон (якщо сильний вітер, то контролер управляє закриттям вікна).
4. Використання *API ключів* (API key – це єдиний безпечний спосіб дати програмі та веб-сайтам доступ до ваших даних) для отримання значень погоди.
5. Зручний додаток на телефон, котрий дає можливість швидко і доступно налаштовувати більшість функцій.

Наявність власної програми розширює можливості використання автоматики, надаючи інструмент з гнучким налаштуванням процесу вирощування рослин з урахуванням індивідуальних вимог. Для певного виду рослин визначені свої параметри роботи (температура, вологість, провітрювання, освітлення), які контролюються і витримуються автоматикою.

Це дозволяє забезпечити максимально комфортні умови процесу вирощування для певного виду рослин, оскільки програми чітко підтримують усі індивідуальні температурні, вологість і повітряно-аераційні параметри процесу. На вимогу замовника можлива зміна кількості, а також параметрів програм роботи гроубокса.

Нижче приводяться зображення індикатора у різних станах (Рис.1 – Рис.6).

Рис.1. Налаштування Температури.

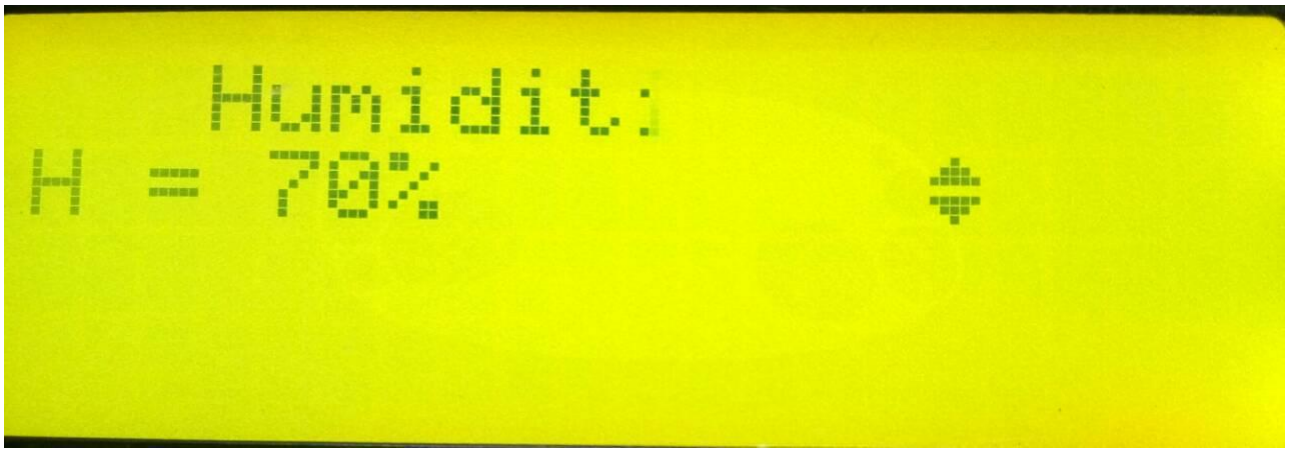

Рис.2. Налаштування Вологості...

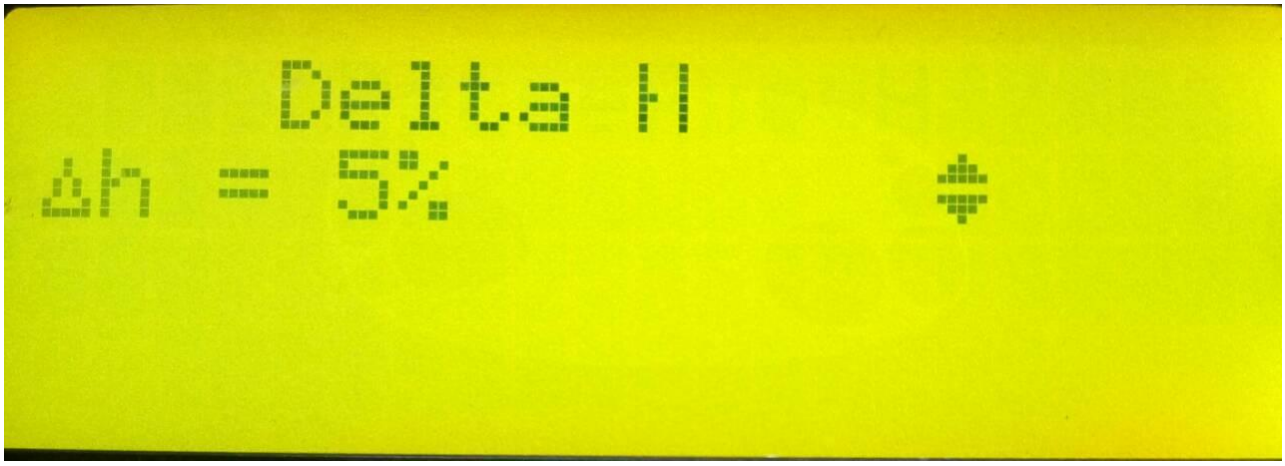

Рис.3. Налаштування Дельти вологості.

Рис.4. Налаштування Таймеру.

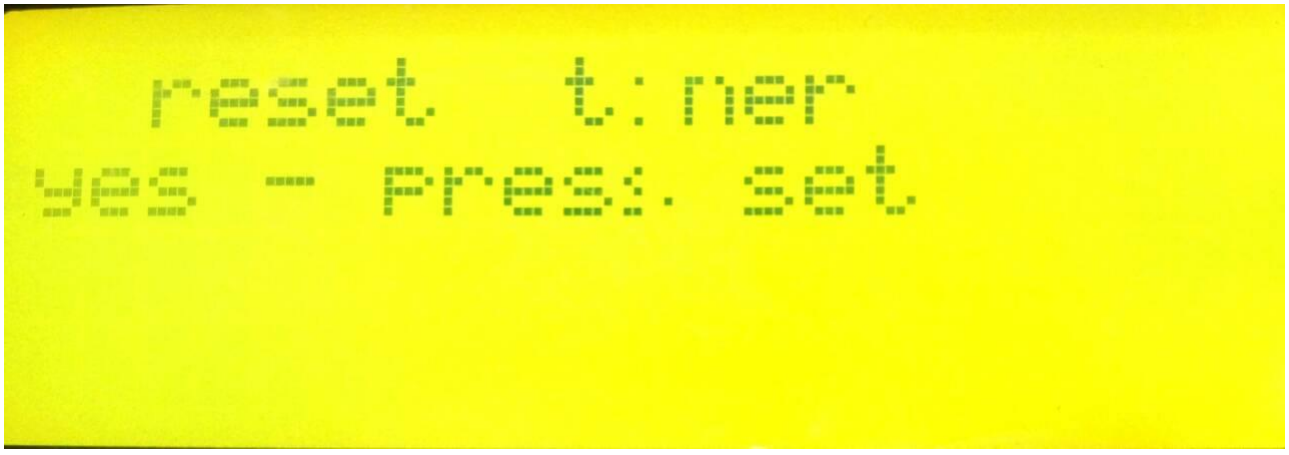

Рис.5. Скидання таймеру.

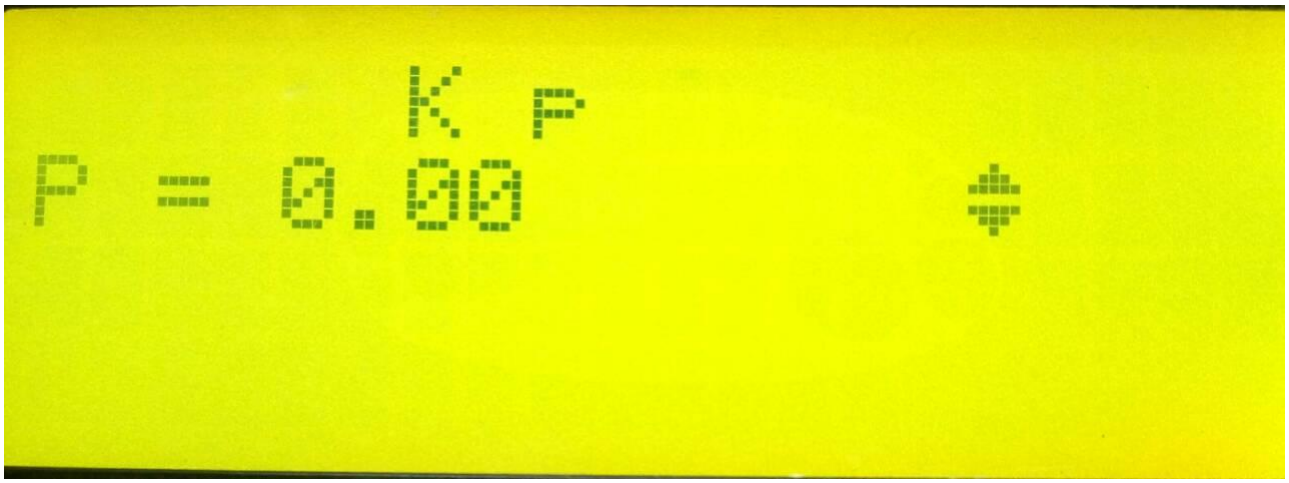

Рис.6. Налаштування Пропорційного коефіцієнту.

Нижче приводяться зображення додатку для смартфону (Рис.7 – Рис.9).

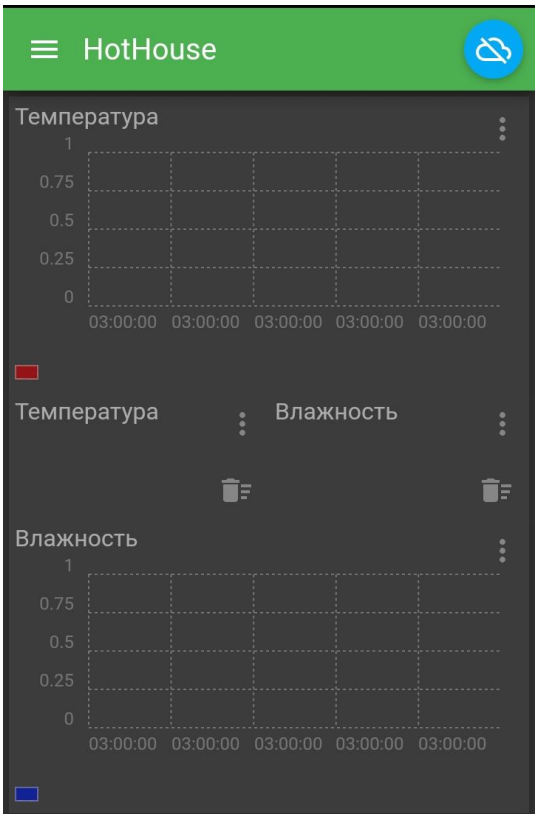

Рис. 7

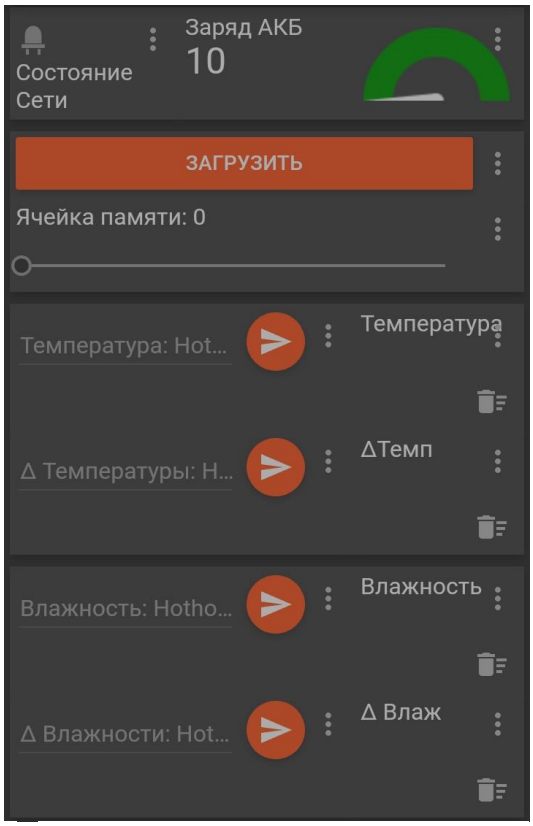

Рис. 8

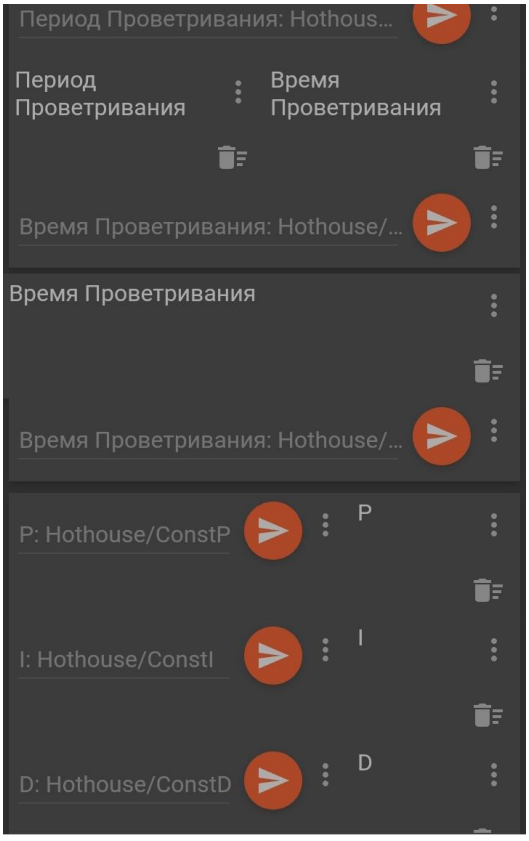

Рис. 9

Програмне забезпечення даного проекту має таку структуру:

- void callback(); *Отримання значень змінних з додатку*
- void setup(); *Попереднє налаштування проекту*
- void RASEEPROMSTimer(); *Перезапуск таймера*
- void SaveToEEPROM(int BankSave); *Збереження даних*
- void LoadFromEEPROM(int BankLoad); *Завантаження даних*
- void PressKeyMenu(); *Опитуваня виду натиснення енкодера*
- void PreSetTime();
- void GoSetTime(int InputX);
- void SetTime();
- void PrintMenuWrite(int FlagM);
- void TimerCalculatePrint(); *Вираховування дати*
- void StartFan(); *Відкриття вікон*
- void StartHot() ; *Керування теном*
- void StartHum(); *Керуванням зволоженням*
- void StartLite(); *Керування світлом*
- void BME280Read(); *Зчитування даних з датчика*
- void SoilMoistureRead(); *Зчитування даних з датчика вологості ґрунту*
- void jsonGet() ; *Отримання даних з сервера*
- void WiFi_funk(); *Відправлення даних на сервер (додаток)*
- void loop(); *Основна функція*

1.2.1. Можливості проекту

Таким чином, гроу-контролер дає змогу змінювати алгоритм роботи в залежності від ваших потреб. Контролер здійснює керування освітленням та

циркуляцією повітря на основі даних про температуру.

Алгоритм роботи блока керування для гроубокса визначається наявністю: обігрівача, зволожувача, механізмом відкриття вікон та освітленням, в залежності від стадії росту рослин та зовнішніх погодних умов.

Система керування кліматом для гроубокса дає змогу підключати виносні датчики: вологості та температури.

Параметри даного контролера для гроубокса, які налаштовуються користувачем:

- Установка реального часу
- Температура нагрівача (вмикання)
- Температура нагрівача (вимикання)
- Тривалість світлового дня
- Тривалість ночі
- Час поливу (сек.)
- Час паузи між поливами (сек.)
- Температура середовища (відкриття вікон)
- Температура середовища (закриття вікон)
- Полив (вмикання/вимикання)
- Налаштування ПІД коефіцієнтів

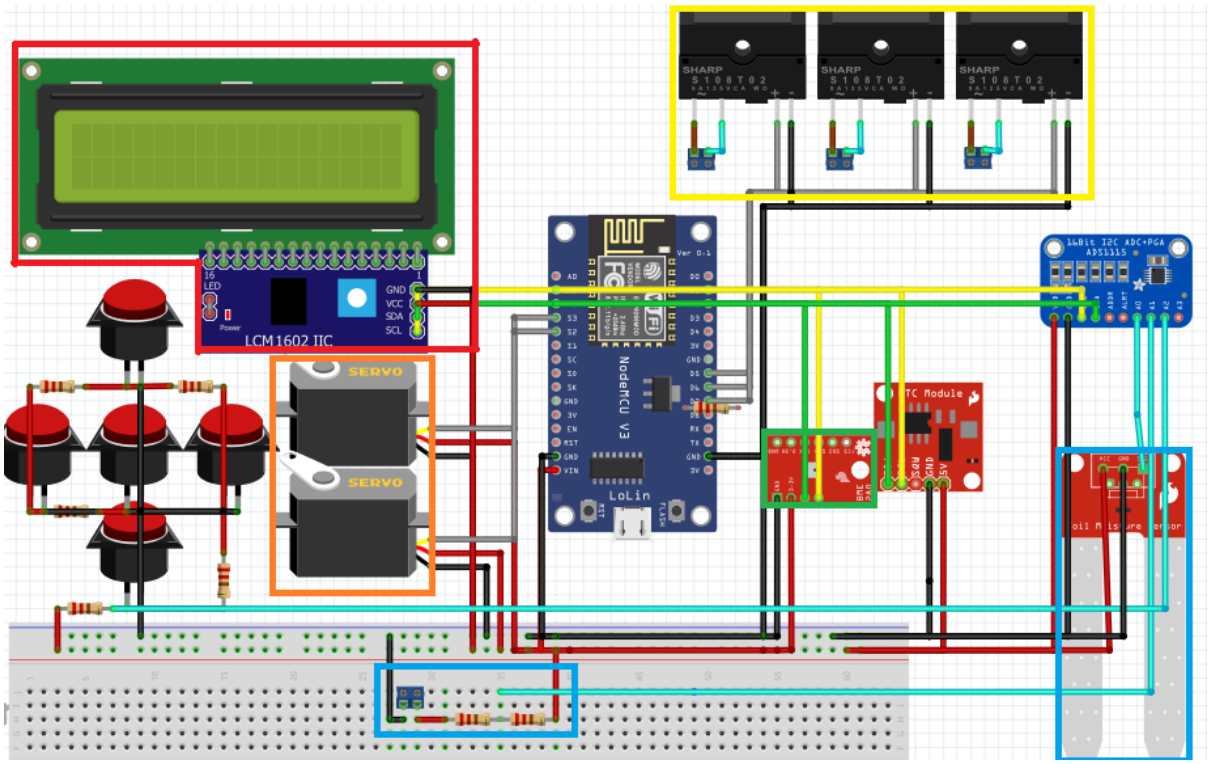

Рис.10. Схема підключення елементів контролера.

На Рис. 10 зображено схему підкючення елементів контролера:

- Синій модуль – це мікроконтроллер Esp8266 NodeMcu v3;
- Блакитний модуль – це плата розшириння ADS1115;
- Зелений модуль – це модуль енкодера призначений для керування контроллера;
- Червоний модуль – це модуль годинника реального часу, призначений для відліку часу навіть тоді, коли немає живлення на контролері;
- Червоній модуль в зеленому прямокутнику – це датчик BME280, призначений для вимірювання температури та вологості;
- Модулі в жовтому прямокутнику – це модулі реле, призначені для керування тен, зволожувачем та подачею води;
- Модулі в помаранчевому прямокутнику – це модулі серво двигуни, котрі відкривають вікна;

- Модулі в червоному прямокутнику – це модулі виводу даних на екран;
- Схема в блакитному прямокутнику – призначена для вимірювання напруги на акумуляторі;
- Модуль в блакитному прямокутнику – призначений для вимірювання вологості ґрунту.

1.2.2. Рекомендації для користувача (інтерфейс)

Дисплей дозволяє відслідковувати і коректувати роботу агрегату в режимі реального часу. Він відображає наступну інформацію:

1. Основні налаштування: температура, вологість, налаштування коефіцієнтів для ПІД регулювання, режим закриття/відкриття вікон.
2. Налаштування виходів реле (нагрівач, вентилятор, компресор підвищення вологості).
3. Збереження налаштування.
4. Завантаження налаштування.
5. Поточний час.
6. Можливість підключення додаткового датчика температур (від 3 і більше).
7. Відображення інформації на LED – дисплей та на «хмарину».
8. Наявність резервного електроживлення.

Автоматизація охоплює практично всі рівні роботи теплиці, будь то полив або провітрювання. Але не варто забувати про недоліки. Одним з недоліків є неправильне налаштування системи, через яку можна втратити весь урожай. Тому до автоматизації потрібно підходити дуже відповідально.

1.2.3. Аналіз отриманих результатів

За допомогою даного проекту можна створити власний програмований гроубокс (міні-теплицю), який має зручний інтерфейс, вдосконалену модель в порівнянні з уже існуючими моделями.

Пропонується автоматизована система контролю клімату в теплиці, що включає в себе все необхідне програмне забезпечення та здійснює повний контроль за процесом вирощування і дозрівання врожаю. Система призначена для автоматизованого контролю параметрів і управління мікрокліматом в тепличному блоці. Використання системи забезпечує високу точність підтримки заданих кліматичних режимів для міні-теплиці за допомогою впливу на виконавчі механізми і обладнання основних технологічних систем і процесів:

– система обігріву повітря;

– система вентиляції;

– система зашторювання;

– система охолодження і зволоження;

– система крапельного зрошення;

– система рециркуляції повітря;

Повністю автоматизована комплексна система вентиляції та контролю, в основу якої покладені правильні розрахунки з урахуванням специфіки конкретних рослин, є запорукою надійності і максимально високих результатів.

2. Висновки

Автоматизація процесу вирощування рослин – це важливий етап розвитку аграрної галузі. Зараз існує великий спектр різноманітних пристроїв для автоматизації практично всіх процесів, що відбуваються в гроубоксі або теплиці. Автоматичні міні-теплиці – це найбільш сучасні пристрої, призначені для догляду за рослинами в домашніх умовах Вони прості в управлінні і при цьому не вимагають додаткової участі людини.

Дотримання інструкції та хороший гроубокс гарантують високий результат. Контролер має досить нескладне меню, на відміну від промислових контролерів. Даний проект має просте управління, нескладну зрозумілу інструкцію, за якою легко налаштовувати контролер. При створенні пристрою ставилось за мету зробити процес вирощування рослин приємним, простим і корисним заняттям. З цією метою і було створено власний програмований гроубокс, який має зручний інтерфейс, вдосконалену модель в порівнянні з уже існуючими моделями.

Електронне управління полегшує роботу, а всі параметри виводяться на дисплей. Цифрові, світлові і графічні індикатори укупі зі звуковою сигналізацією дозволяють відслідковувати процеси, що відбуваються, в режимі реального часу.

Проект досить простий в користуванні, цікавий і зрозумілий. Він може бути корисний кожному, хто хоче створити хорошу власну міні-теплицю та мати гарантовано високий результат.

Список використаних джерел

1. М. Эллис, Б. Строуструп. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992. 445с.
2. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.
3. https:// http://boteon.com/blogs/verit-buduschemu/-robototehn-ka-elektron-ka-programuvannja-kontroler-v.html Робототехніка. Електроніка. Програмування контролерів.
4. https://pikabu.ru/story/universalnyiy_kontroller_dlya_teplits_grouboksov_inkubatorov_6698418 – Контролер розумної теплиці своїми руками.
5. http://www.programmer-lib.ru/csharp.php – уроки по C++.
6. http://programer.in.ua/index.php/uroky/uroky-dlia-vyvchennia-c – програмування на C#.

Додатки

Додаток 1.

Код програми

```
#include <OneWire.h>
#include <DS1307new.h>
#include <GyverEncoder.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <Servo.h>
#include <EEPROM.h>
#include <FastIO.h>
#include <I2CIO.h>
#include <LCD.h>
#include <LiquidCrystal.h>
#include <LiquidCrystal_SR.h>
#include <LiquidCrystal_I2C.h>
#include <LiquidCrystal_SR2W.h>
#include <LiquidCrystal_SR3W.h>
#define EEPROM_ADDR 0x50
#include <PID_v1.h>
#include <ArduinoJson.h>
#include <MQTT.h>
#include <PubSubClient.h>
#include <PubSubClient_JSON.h>
#include <ESP8266WiFi.h>
#include <stdlib.h>
#include <Time.h>
```

```
#include <TimeLib.h>
#include <sunMoon.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>


// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).

#define ssid  "Kyivstar_Alpha"  // Имя вайфай точки доступа
#define pass  "R667fksX55fr" // Пароль от точки доступа

#define mqtt_server "soldier.cloudmqtt.com" // Имя сервера MQTT
#define  mqtt_port 12984 // Порт для подключения к серверу MQTT
#define mqtt_user "vvhzufvr" // Логи от сервер
#define mqtt_pass "yKN0yiP31mkt" // Пароль от сервера
#define mqtt_id "HotHouse" // Пароль от сервера

#define host "api.openweathermap.org"
#define wapi  "GET /data/2.5/weather?q=Kyiv,ua&appid=c557d50e325f16c912651e1054527de7 HTTP/1.1"
#define httpPort  80

#define OUR_latitude    55.751244     // координаты Киева
#define OUR_longtitude  37.618423
#define OUR_timezone    120                   // localtime with UTC difference in minutes
sunMoon  sm;

#define SDA_PIN 4
#define SCL_PIN 5
```

```
String line;
int NumBankSave = 0;
int IfBankSave  = 0;

Servo rservo;
Servo lservo;

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  //LiquidCrystal lcd(11, 10, 9, 8, 7, 6);   инициализация входов на дисплей

WiFiClient wclient; // Use WiFi_funkClient class to create TCP connections
PubSubClient client(wclient, mqtt_server, mqtt_port);

Adafruit_BME280 bme;

Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */

// Must declare output stream before Adafruit_ESP8266 constructor; can be
// a SoftwareSerial stream, or Serial/Serial1/etc. for UART.
Encoder Enc(13, 15, 12, TYPE2);                 // CLK, DT, SW, тип (TYPE1 / TYPE2): TYPE1 одношаговый, TYPE2 двухшаговый. Если ваш энкодер работает странно, смените тип
int BS, BL, BankSave;                                   // Переменые для сохранения настроек
double Temp = 37.7, deltaT = 0.2;               // температура выращивания, дельта Т
#define maxdeltaT 2                               // максимальное значение дельтаТ для меню
#define mindeltaT 0.1                             // минимальное  значение дельтаТ для меню
double maxTempFanStart = 37.9;                          // максимально допустимая температура при которой включается аварийная продувка
#define maxTempFanStartMenuMin 16                       // максимальное значение температуры работы вентилятора для меню
#define maxTempFanStartMenuMax 61                       // минимальное значение температуры работы вентилятора для меню

int TimeFanWork = 60;                                   // время проветривания теплицы от CO2 в секундах
```

```
#define maxTimeFanWork 900                          // максимальное время проветривания теплицы от CO2 для меню в секундах
#define minTimeFanWork 5                            // минимальное время проветривания теплицы от CO2 для меню в секундах
#define FadeTimeFanWork 1                           // шаг изменения значения времени работы вентилятора в меню в секундах
int TimeIntervalFanWork = 30;                       // интервал между включениями проветривания теплицы от CO2 в минутах
#define maxTimeIntervalFanWork 300                  // максимальный интервал между включениями проветривания теплицы от CO2
для меню в минутах
#define minTimeIntervalFanWork 1                    // минимальный интервал между включениями проветривания теплицы от CO2
для меню в минутах
#define FadeTimeIntervalFanWork 1                   // шаг изменения значения интервала между включениями проветривания
теплицы от CO2 в меню в минутах
int FanWorkFlag = 1;                        // флаг активности продувки теплицы от CO2

#define maxTempDanger 70                    // максимальная температура выращивания для меню
#define minTempDanger 20                    // минимальная температура выращивания для меню
#define FadeAmountTemp 0.1                  // шаг изменения тепертуры в меню
int Humiditi = 60, deltaHumiditi = 1;       // влажность выращивания дельта влажности
int HumGroundnow,HumGround = 60;            // влажность почвы и тип теплицы
#define mindeltaHum 1                       // минимальная дельта влажности для меню
#define maxdeltaHum 10                      // максимальная дельта влажности для меню
#define FadeAmountdeltaHum 1                // шаг изминения дельты влажности для меню
#define FadeAmountHum 1                     // шаг изменения влажности в меню
#define MaximumHumiditi 100                 // максимальное значение влажности для меню
#define MinimumHumiditi 0                   // минимальное значение влажности для меню

float tempK,tempC,tempKmin,tempCmin,tempKmax,tempCmax ;     // достаем температуру из структуры main

int pressurehPa;        // достаем давление из структуры main
float pressure;         // давление окружения
double Press ;          // давление внутри
int humidity ;          // достаем влажность из структуры main
```

```
float windspeed;          // достаем скорость ветра из структуры main
int winddeg;              // достаем направление ветра из структуры main



int i = 0; int k = 0;
double Tnow;                                  // Реальная температура на данный момент в теплице на BME280
int hum;                                      // Реальная влажность на данный момент в теплице на BME280
int TypeHouse = 1;
int VarHouse[3] ={0,1,2};
int MainMenu = 0, SubMenu = 0, FlagMenu = 0;                    // переменные для управления меню
int NOWyear, NOWmonth, NOWday, NOWhour, NOWminute, NOWsecond;
int Setyear, Setmonth, Setday, Sethour, Setminute, Setsecond;
bool TIFlagd = 0, TIFlagf = 0;                                  // флаг индикации таймера выращивания 0 - таймер не активен, 1 - таймер активен
int TRyear, TRmonth, TRday, TRhour, TRminute, TRsecond;         // переменные в которых будет сохранена дата начала выращивания
int bank = 0;                                 // номер банка записи и загрузки настроек выращивания
int FlagTempFan;
int FlagTimeFan = 0;    //
//Define Variables we'll be connecting to
double Output;
//Define the aggressive and conservative Tuning Parameters
double consKp=1, consKi=0.05, consKd=0.25;
PID myPID(&Tnow, &Output, &Temp, consKp, consKi, consKd, DIRECT);  //Specify the links and initial tuning parameters

uint8_t strelka_vverh_vniz[8] = { B00100, B01110, B11111, B00000, B11111, B01110, B00100 };     // закодирована в двоичной системе
СТРЕЛКА ВВЕРХ ВНИЗ
uint8_t temp_cel[8]       = { B11000, B11000, B00110, B01001, B01000, B01001, B00110 };     // закодирована в двоичной системе ГРАДУС
ЦЕЛЬСИЯ
uint8_t temp_del[8]       = { B00000, B00100, B00100, B01010, B01010, B10001, B11111 };     // закодирована в двоичной системе ДЕЛЬТА
```

```
uint8_t   Hot_ON[8]          = { B01111, B10000, B00111, B00001, B01110, B10000, B01111 };     // закодирована в двоичной системе ОБОГРЕВ
uint8_t   Fan_ON[8]          = { B11011, B11011, B11011, B00100, B00100, B11011, B11011 };     // закодирована в двоичной системе ОБДУВ
uint8_t   Hum_ON[8]           = { B00100, B00100, B01110, B11011, B10111, B10111, B01110 };     // закодирована в двоичной системе УВЛАЖНЕНИЕ
uint8_t   WiFi_funk_ON[8]     = { B00000, B11111, B01110, B10101, B01110, B00100, B00000 };     // закодирована в двоичной системе WiFi_funk



#define PinHot  0                       // первоначальное подключение обогрева на реле №1
#define PinLite 2                       // первоначальное подключение обдува на реле №2
#define PinHum  14                          // первоначальное подключение увлажнителя на реле №3
#define PinHumGroundControl 10                  // первоначальное подключение датчика влажности земли на пин 10
#define PinHumGround 17
float voltage ;           // напряжение на аккумуляторе
int netpower ;          // наличие сети

int m;                           // значения отображения информации на табле
int buttons_Menu;            // значения кнопок меню
int PressingButtons;

unsigned long int currentTime = 0 ;
unsigned long int loopTime = 0 ;

unsigned long int currentMillis;
unsigned long int StartMillis = 0 ;                                 //счетчик прошедшего времени для AutoStartMenu
#define interval 15000                                  //задержка автовозврата к StartMenu 7сек
unsigned long int BME280readMillis = 0 ;                                    //счетчик прошедшего времени для интервала чтения с датчика температуры
```

```
#define BME280interval 100                          //опрос датчика температуры каждую 0.1 сек
unsigned long int I2CGroundreadMillis = 0 ;                          //счетчик прошедшего времени для интервала чтения с датчика температуры
#define I2CGroundinterval 250                       //опрос датчика температуры каждую 0.25 сек
unsigned long int MenuRewriteMillis = 0 ;                          //счетчик прошедшего времени для обновления главного меню
#define MenuRewriteinterval 500                     //обновление главного меню через 0,5сек
unsigned long int WiFi_funkwriteMillis = 0;
unsigned long int TimeFaning = 0;                          //счетчик прошедшего времени для продувки теплицы от CO2
unsigned long int TimeIntervalFaning;                          //счетчик прошедшего времени для интервала между продувками теплицы от CO2
unsigned long int TimeFaningInterval = TimeFanWork * 1000;          //длительность работы вентилятора при продувке теплицы от CO2
unsigned long int TimeIntervalFaningInterval = TimeIntervalFanWork * 60000;//длительность интервала между продувками теплицы от CO2

void setup()
{
  lcd.begin(20, 4);
  Wire.begin();
  Serial.begin(115200);          // Запускаем вывод данных на серийный порт
  Wire.begin(SDA_PIN, SCL_PIN);

  ads.begin();

  rservo.attach(0);
  lservo.attach(2);
  myPID.SetMode(AUTOMATIC);
  Serial.println();

  Serial.print("Connecting to ");
  Serial.println(ssid);
```

```
 WiFi.begin(ssid, pass);
Serial.print("checking wifi...");
 while (WiFi.status() != WL_CONNECTED) {
   Serial.print(".");
   delay(1000);
 }

  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

 loopTime = currentTime;

 pinMode(PinHot, OUTPUT);                    // Реле №1 на 2 выходе
 pinMode(PinLite, OUTPUT);                   // Реле №2 на 3 выходе
 pinMode(PinHum, OUTPUT);                     // Реле №3 на 4 выходе
 pinMode(PinHumGroundControl, OUTPUT);               // Управление датчиком на 5 выходе

 byte lowByte = EEPROM.read(0); delay(100);                                  // чтение lowByte температуры выращивания *100 из ячейки "0"
 byte higtByte = EEPROM.read(1); delay(100);                                 // чтение higtByte температуры выращивания *100 из ячейки "1"
 Temp = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
 lowByte = EEPROM.read(2); delay(100);                                       // чтение lowByte дельтаТ температуры выращивания *100 из ячейки
"2"
 higtByte = EEPROM.read(3); delay(100);                                      // чтение higtByte дельтаТ температуры выращивания *100 из ячейки
"3"
 deltaT = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
 Humiditi = EEPROM.read(4); delay(100);                                  // чтение дельты влажности выращивания из ячейки "4"
 deltaHumiditi = EEPROM.read(5);  delay(100);                              // чтение дельты влажности выращивания из ячейки "5"

 lowByte = EEPROM.read(7);  delay(100);                                      // чтение lowByte температуры продувки *100 из ячейки "7"
```

```
  higtByte = EEPROM.read(8);  delay(100);                                  // чтение higtByte температуры продувки *100 из ячейки "8"
  maxTempFanStart = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
  lowByte = EEPROM.read(9);  delay(100);                                   // чтение lowByte времени вентиляции теплицы от CO2 из ячейки "9"
  higtByte = EEPROM.read(10); delay(100);                                  // чтение higtByte времени вентиляции теплицы от CO2 из ячейки "10"
  TimeFanWork = ((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00);
  lowByte = EEPROM.read(11); delay(100);                                   // чтение lowByte интервала между вентиляциями теплицы от CO2 из ячейки "11"
  higtByte = EEPROM.read(12); delay(100);                                  // чтение higtByte интервала между вентиляциями теплицы от CO2 из ячейки "12"
  TimeIntervalFanWork = ((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00);
  FanWorkFlag = EEPROM.read(13); delay(100);                               // чтение флага активностивентиляциями теплицы от CO2 из ячейки "13"
  TimeFaningInterval = TimeFanWork * 1000;                                 //чтение длительности работы вентилятора при продувке теплицы от CO2
  TimeIntervalFaningInterval = TimeIntervalFanWork * 60000;                //чтение длительности интервала между продувками теплицы от CO2
  consKp = EEPROM.read(131); delay(100);                                   //чтение Пропорционального коефициента
  consKi = EEPROM.read(132); delay(100);                                   //чтение Интегрального коефициента
  consKd = EEPROM.read(133); delay(100);                                   //чтение Дефиринциального коефициента
  HumGround = EEPROM.read(114); delay(100);                                //чтение Нужной вдоажности почвы
  TypeHouse = EEPROM.read(135); delay(100);                                //чтение Типа теплицы

                                  // ---------- чтение из EEPROM установок настройки теплицы сохраненных в банк настроек №1
                                 // ---------- чтение из EEPROM установок и времени начала выращивания таймера выращивания
  TRyear = EEPROM.read(120) + 2000; delay(100);                            // чтение года начала выращивания
  TRmonth = EEPROM.read(121); delay(100);                                  // чтение месяца начала выращивания
  TRday = EEPROM.read(122); delay(100);                                    // чтение дня начала выращивания
  TRhour = EEPROM.read(123); delay(100);                                   // чтение часа начала выращивания
  TRminute = EEPROM.read(124); delay(100);                                 // чтение минуты начала выращивания
```

```
  TRsecond = EEPROM.read(125); delay(100);                                  // чтение секунды начала выращивания
  TIFlagd = EEPROM.read(126); delay(100);                                   // чтение флага ВКЛЮЧЕН (1) / ВЫКЛЮЧЕН (0)  индикации таймера выращивания
  TIFlagf = EEPROM.read(127); delay(100);                                   // чтение флага ВКЛЮЧЕН (1) / ВЫКЛЮЧЕН (0)  таймера выращивания
                                  // ---------- чтение из EEPROM установок и времени начала выращивания таймера выращивания

                          // ---------- чтение из EEPROM установок настройки теплицы сохраненных в банк настроек №0

  lcd.createChar(1, strelka_vverh_vniz);  lcd.createChar(3, temp_cel);  lcd.createChar(4, temp_del);  lcd.createChar(5, Hot_ON);  lcd.createChar(6, Fan_ON);  lcd.createChar(7, Hum_ON); lcd.createChar(8, WiFi_funk_ON);

  lcd.setCursor(0, 0);  lcd.print("*-*-*-*-****-*-*-*-*");
  lcd.setCursor(0, 1);  lcd.print("    AutoHothouse    ");
  lcd.setCursor(0, 2);  lcd.print("  By_Vurchun_V-0.6  ");
  lcd.setCursor(0, 3);  lcd.print("*-*-*-*-****-*-*-*-*");
  delay(1500);
  lcd.clear();
  currentTime = millis();
  loopTime = currentTime;

}
void RASEEPROMSTimer()
{
  TRyear = RTC.year - 2000; TRmonth = RTC.month; TRday = RTC.day; TRhour = RTC.hour; TRminute = RTC.minute; TRsecond = RTC.second;
  EEPROM.write(120, TRyear); delay(100);                                    // запись года начала выращивания в ячейку 120 EEPROM
  EEPROM.write(121, TRmonth); delay(100);                                   // запись месяца начала выращивания в ячейку 121 EEPROM
  EEPROM.write(122, TRday); delay(100);                                     // запись дня начала выращивания в ячейку 122 EEPROM
  EEPROM.write(123, TRhour); delay(100);                                    // запись часа начала выращивания в ячейку 123 EEPROM
```

```
  EEPROM.write(124, TRminute); delay(100);                                  // запись минуты начала выращивания в ячейку 124 EEPROM
  EEPROM.write(125, TRsecond); delay(100);                                  // запись секунды начала выращивания в ячейку 125 EEPROM
}

void SaveToEEPROM(int BankSave)                                             // запись данных во внутренний EEPROM
{
  int TIC = int(Temp * 100); byte lowByte = ((TIC >> 0) & 0xFF); byte higtByte = ((TIC >> 8) & 0xFF);
  BS = BankSave * 20 + 0;     EEPROM.write(BS, lowByte);                delay(100);    // запись lowByte температуры выращивания * 100 в ячейку "0" банка "bank"
  BS = BankSave * 20 + 1;     EEPROM.write(BS, higtByte);               delay(100);    // запись higtByte температуры выращивания * 100 в ячейку "1" банка "bank"
  int DTIC = int(deltaT * 100);  lowByte = ((DTIC >> 0) & 0xFF); higtByte = ((DTIC >> 8) & 0xFF);
  BS = BankSave * 20 + 2;     EEPROM.write(BS, lowByte);                delay(100);    // запись lowByte дельтаТ температуры выращивания * 100 в ячейку "2" банка "bank"
  BS = BankSave * 20 + 3;     EEPROM.write(BS, higtByte);               delay(100);    // запись higtByte дельтаТ температуры выращивания * 100 в ячейку "3" банка "bank"
  BS = BankSave * 20 + 4;     EEPROM.write(BS, Humiditi);       delay(100);    // запись дельты влажности выращивания в ячейку "4" банка "bank"
  BS = BankSave * 20 + 5;     EEPROM.write(BS, deltaHumiditi);  delay(100);    // запись дельты влажности выращивания в ячейку "5" банка "bank"

  int TICc = int(maxTempFanStart * 100); lowByte = ((TICc >> 0) & 0xFF); higtByte = ((TICc >> 8) & 0xFF);
  BS = BankSave * 20 + 7;     EEPROM.write(BS, lowByte);                delay(100);    // запись lowByte температуры продувки * 100 в ячейку "7" банка "bank"
  BS = BankSave * 20 + 8;     EEPROM.write(BS, higtByte);               delay(100);    // запись higtByte температуры продувки * 100 в ячейку "8" банка "bank"
  lowByte = ((TimeFanWork >> 0) & 0xFF); higtByte = ((TimeFanWork >> 8) & 0xFF);
  BS = BankSave * 20 + 9;     EEPROM.write(BS, lowByte);                delay(100);    // запись lowByte времени вентиляции теплицы от CO2 в ячейку "9" банка "bank"
  BS = BankSave * 20 + 10;    EEPROM.write(BS, higtByte);               delay(100);    // запись higtByte времени вентиляции теплицы от CO2 в
```

```
ячейку "10" банка "bank"
  lowByte = ((TimeIntervalFanWork >> 0) & 0xFF); higtByte = ((TimeIntervalFanWork >> 8) & 0xFF);
  BS = BankSave * 20 + 11;    EEPROM.write(BS, lowByte);                delay(100);    // запись lowByte интервала между вентиляциями
теплицы от CO2 в ячейку "11" банка "bank"
  BS = BankSave * 20 + 12;    EEPROM.write(BS, higtByte);               delay(100);    // запись higtByte интервала между вентиляциями
теплицы от CO2 в ячейку "12" банка "bank"
  BS = BankSave * 20 + 13;    EEPROM.write(BS, FanWorkFlag);            delay(100);    // запись флага активностивентиляциями теплицы от
CO2 в ячейку "13" банка "bank"
  BS = BankSave * 20 + 14;    EEPROM.write(BS, HumGround);              delay(100);    // запись Нужной вдоажности почвы в ячейку "14"
банка "bank"

 }
void LoadFromEEPROM(int BankLoad)                                       // загрузка данных из внутреннего EEPROM
{
  BL = BankLoad * 20 + 0; byte lowByte = EEPROM.read(BL);           delay(100);    // чтение lowByte температуры выращивания *100 из
ячейки "0"
  BL = BankLoad * 20 + 1;     byte higtByte = EEPROM.read(BL);          delay(100);    // чтение higtByte температуры выращивания *100 из
ячейки "1"
  Temp = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
  BL = BankLoad * 20 + 2;     lowByte = EEPROM.read(BL);                delay(100);    // чтение lowByte дельтаТ температуры выращивания
*100 из ячейки "2"
  BL = BankLoad * 20 + 3;     higtByte = EEPROM.read(BL);               delay(100);    // чтение higtByte дельтаТ температуры выращивания
*100 из ячейки "3"
  deltaT = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
  BL = BankLoad * 20 + 4;     Humiditi = EEPROM.read(BL);       delay(100);    // чтение дельты влажности выращивания из ячейки "4"
  BL = BankLoad * 20 + 5;     deltaHumiditi = EEPROM.read(BL);  delay(100);    // чтение дельты влажности выращивания из ячейки "5"

  BL = BankLoad * 20 + 7;     lowByte = EEPROM.read(BL);                delay(100);    // чтение lowByte температуры продувки *100 из ячейки
"7"
  BL = BankLoad * 20 + 8;     higtByte = EEPROM.read(BL);               delay(100);    // чтение higtByte температуры продувки *100 из ячейки
```

```
"8"
  maxTempFanStart = (((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00)) / 100.00;
  BL = BankLoad * 20 + 9;     lowByte = EEPROM.read(BL);  delay(100);                 // чтение lowByte времени вентиляции теплицы от CO2 из ячейки "9"
  BL = BankLoad * 20 + 10;    higtByte = EEPROM.read(BL);  delay(100);                // чтение higtByte времени вентиляции теплицы от CO2 из ячейки "10"
  TimeFanWork = ((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00);
  BL = BankLoad * 20  + 11;    lowByte = EEPROM.read(BL); delay(100);                 // чтение lowByte интервала между вентиляциями теплицы от CO2 из ячейки "11"
  BL = BankLoad * 20 + 12;    higtByte = EEPROM.read(BL); delay(100);                 // чтение higtByte интервала между вентиляциями теплицы от CO2 из ячейки "12"
  TimeIntervalFanWork = ((lowByte << 0) & 0xFF) + ((higtByte << 8) & 0xFF00);
  BL = BankLoad * 20 + 13;    FanWorkFlag = EEPROM.read(BL); delay(100);              // чтение флага активностивентиляциями теплицы от CO2 из ячейки "13"
  BL = BankLoad * 20 + 14;    HumGround = EEPROM.read(BL); delay(100);                // чтение Нужной вдоажности почвы в ячейку "14" банка "bank"

  TimeFaningInterval = TimeFanWork * 1000;                //длительность работы вентилятора при продувке теплицы от CO2
  TimeIntervalFaningInterval = TimeIntervalFanWork * 60000;//длительность интервала между продувками теплицы от CO2
}


void PressKeyMenu()                                       // Вычиление нажатия кнопок
{
  delay(50);

  PressingButtons = 0;

  buttons_Menu = ads.readADC_SingleEnded(0);
  Serial.print("  Resistant key button module="); Serial.print(buttons_Menu); Serial.println(" ");
```

```
      if (buttons_Menu >= 60000 )  PressingButtons = 1;          // меню
  else  if (buttons_Menu > 10000 && buttons_Menu < 13000) PressingButtons = 2;     // вверх
  else  if (buttons_Menu > 3000 && buttons_Menu < 7000) PressingButtons = 3;     // вниз
  else  if ( buttons_Menu > 15000 && buttons_Menu < 17000)PressingButtons = 4;      // выбор
  else  if (buttons_Menu > 800 && buttons_Menu < 900) PressingButtons = 5;  // открития окон
  else  PressingButtons = 0;
 Serial.print("  Button key =");Serial.print(PressingButtons); Serial.println(" ");delay(50);
}

void PreSetTime()
{
  Setyear = RTC.year; Setmonth = RTC.month; Setday = RTC.day; Sethour = RTC.hour; Setminute = RTC.minute; Setsecond = RTC.second;
}
void GoSetTime(int InputX){
  switch (SubMenu) {
  case 1:  Setyear   = InputX;    break;
  case 2:  Setmonth  = InputX;   break;
  case 3:  Setday    = InputX;     break;
  case 4:  Sethour   = InputX;    break;
  case 5:  Setminute = InputX;  break;
  case 6:  Setsecond = InputX;  break;
   }
}

void SetTime()
{
  RTC.getTime();
  RTC.stopClock();
  RTC.fillByYMD(Setyear, Setmonth,  Setday   );  delay(250);
  RTC.fillByHMS(Sethour, Setminute, Setsecond);  delay(250);
```

```
  RTC.setTime();
  delay(100);
  RTC.startClock();
}
void PrintMenuWrite(int FlagM)
{
  switch (FlagM) {
  case 0:  lcd.setCursor(15, 1);  lcd.print("\1");  break;
  case 1:  lcd.setCursor(15, 1);  lcd.print("*" );   break;
  }
}

int Sec, SecPer, Min, MinPer, Hou, HouPer, Dey, DeyPer, Mon, MonPer, Yer;
void TimerCalculatePrint()
{
  if (TIFlagf == 1) {
    if (RTC.second >= TRsecond) { Sec = RTC.second - TRsecond;          SecPer = 0; }
    else { Sec = 60 + (RTC.second - TRsecond);          SecPer = 1; }
    if ((RTC.minute - SecPer) >= TRminute) { Min = RTC.minute - TRminute - SecPer; MinPer = 0; }
    else { Min = 60 + (RTC.minute - TRminute) - SecPer; MinPer = 1; }
    if ((RTC.hour - MinPer) >= TRhour) { Hou = RTC.hour - TRhour - MinPer; HouPer = 0; }
    else { Hou = 24 + (RTC.hour - TRhour) - MinPer; HouPer = 1; }
    if ((RTC.day - HouPer) >= TRday) { Dey = RTC.day - TRday - HouPer; DeyPer = 0; }
    else {
      switch (RTC.month) {
      case 2: { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
      case 3: {
        if (RTC.year % 4 == 0 && RTC.year % 100 != 0 || RTC.year % 400 == 0)
        {
          Dey = 29 + (RTC.day - TRday) - HouPer; DeyPer = 1;
```

```
      }
      else
      {
        Dey = 28 + (RTC.day - TRday) - HouPer; DeyPer = 1;
      }
      break; }
    case 4:  { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 5:  { Dey = 30 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 6:  { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 7:  { Dey = 30 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 8:  { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 9:  { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 10: { Dey = 30 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 11: { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 12: { Dey = 30 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    case 1:  { Dey = 31 + (RTC.day - TRday) - HouPer; DeyPer = 1; break; }
    }
  }
  if ((RTC.month - DeyPer) >= TRmonth) { Mon = RTC.month - TRmonth - DeyPer; MonPer = 0; }
  else { Mon = 12 + (RTC.day - TRmonth) - DeyPer; MonPer = 1; }
  Yer = RTC.year - TRyear;
 }
 }



void StartFan() // Открытие и закрытие окон и печать их состояний на дисплей
{
 if (Tnow >= (maxTempFanStart + (deltaT / 2)))
 {
```

```
   FlagTempFan = 1;
  }
  else
  {
   if (Tnow < (maxTempFanStart - (deltaT / 2)))
   {
    FlagTempFan = 0;
   }
  }
  if (FanWorkFlag == 1) {
   switch (FlagTimeFan) {
   case 1: {
    if ((currentMillis - TimeFaning) > TimeFaningInterval) {
     FlagTimeFan = 0;
     TimeIntervalFaning = currentMillis;
    } break; }
   case 0: {
    if ((currentMillis - TimeIntervalFaning) > TimeIntervalFaningInterval) {
     FlagTimeFan = 1;
     TimeFaning = currentMillis;
    } break; }
   }
  }
      if (FlagTempFan == 1 || FlagTimeFan == 1 && windspeed < 15) { rservo.write(180); lservo.write(180);lcd.setCursor(19, 2); lcd.print("\6"); }
  else if (FlagTempFan == 1 || FlagTimeFan == 1 && windspeed < 10 && winddeg < 45  && winddeg > 135) { rservo.write(0); rservo.write(90);
lcd.setCursor(19, 2); lcd.print("\6"); }
  else if (FlagTempFan == 1 || FlagTimeFan == 1 && windspeed < 10 && winddeg > 315 && winddeg < 225) { lservo.write(0); lservo.write(90);
lcd.setCursor(19, 2); lcd.print("\6"); }
  else { rservo.write(0); rservo.write(0); lcd.setCursor(19, 2); lcd.print(" "); }
}
```

```
void TypeHousePrint(){
   switch(TypeHouse){
   case 0:lcd.setCursor(8, 1);lcd.print("earthen");
   case 1:lcd.setCursor(8, 1);lcd.print("hidroponik");
   case 2:lcd.setCursor(8, 1);lcd.print("aeroponik");
   }
}

void BME280Read()                                    // Чтение значений с датчика BME280
{
  Serial.println(F("BME280 test"));
   bool status;
     status = bme.begin();
   if (!status) {
       Serial.println("Could not find a valid BME280 sensor, check wiring!");
       while (1);
   }

   Serial.println("-- Default Test --");


  if (currentMillis - BME280readMillis > BME280interval)
  {
   BME280readMillis = currentMillis;
   Tnow = bme.readTemperature();
   Press = bme.readPressure() / 100.0F;
   hum = bme.readHumidity();
 }
}
void SoilMoistureRead(){
```

```
  if (currentMillis - I2CGroundreadMillis > I2CGroundinterval)
  {
  I2CGroundreadMillis = currentMillis;
  digitalWrite(PinHumGroundControl, HIGH);
  HumGroundnow = ads.readADC_SingleEnded(2);
  HumGroundnow = map(HumGroundnow, 0, 1000, 0, 100);  // адаптируем значения от 0 до 100,

  digitalWrite(PinHumGroundControl, LOW);

 }
}

void StartHot()                                  // включение и отключение обогрева и печать их состояний на дисплей
{
  if (Tnow < (Temp - (deltaT / 2)))
  {
  myPID.SetTunings(consKp, consKi, consKd);
  myPID.Compute();
  analogWrite(PinHot, Output);
  lcd.setCursor(19, 0);
  lcd.print("\5");

}
  else
  {
    if (Tnow >= (Temp + (deltaT / 2)))
    {
      digitalWrite(PinHot, HIGH);
      if (Tnow < (Temp + (deltaT / 2)))
      {
```

```
        lcd.setCursor(19, 0);
        lcd.print(" ");
       }
     }
   }
}
void StartHum()
{
  if (hum < (Humiditi - (deltaHumiditi / 2)))
  {
    digitalWrite(PinHum, LOW);
    lcd.setCursor(19, 1);
    lcd.print("\7");

  }
  else
  {
    if (hum >(Humiditi + (deltaHumiditi / 2)))
    {
      digitalWrite(PinHum, HIGH);
      lcd.setCursor(19, 1);
      lcd.print(" ");

    }
  }
   if (TypeHouse == 0){
    SoilMoistureRead();
 if (HumGroundnow < (HumGround - (deltaHumiditi / 2)))
  {
    digitalWrite(PinHum, LOW);
```

```
   lcd.setCursor(19, 1);
   lcd.print("\7");

  }
  else
  {
   if (HumGroundnow >(HumGround + (deltaHumiditi / 2)))
   {
    digitalWrite(PinHum, HIGH);
    lcd.setCursor(19, 1);
    lcd.print(" ");

   }
  }
 }
}
void StartLite()
{
  tmElements_t  tm;                          // specific time
  sm.init(OUR_timezone, OUR_latitude, OUR_longtitude);
  time_t sRise = sm.sunRise();
  time_t sSet  = sm.sunSet();

  if (sRise == RTC.hour,RTC.minute)
  {
   digitalWrite(PinHum, LOW);
   lcd.setCursor(19, 1);
   lcd.print("\7");

  }
```

```
  else
  {
    if (sSet == RTC.hour,RTC.minute)
    {
      digitalWrite(PinHum, HIGH);
      lcd.setCursor(19, 1);
      lcd.print(" ");

    }
  }
}

void callback(const MQTT::Publish& pub) {
  Serial.print(pub.topic());
  Serial.print(" => ");
    uint8_t buf[100];
    int read;
  if (pub.has_stream()) {

    while (read = pub.payload_stream()->read(buf, 100)) {
      Serial.write(buf, read);
    }
    pub.payload_stream()->stop();
    Serial.println("");
  } else
  if(String(pub.topic()) == "Hothouse/ControlHum")                { Humiditi = pub.payload_string().toInt(); }// проверяем из нужного ли нам
топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/ControlTemp")                { Temp = pub.payload_string().toInt(); }// проверяем из нужного ли нам топика
пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/TimeIntervalFanWorkntrolTemp") { TimeIntervalFanWork = pub.payload_string().toInt(); }// проверяем из
```

```
нужного ли нам топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/TimeFanWork")                { TimeFanWork = pub.payload_string().toInt(); }// проверяем из нужного ли
нам топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/maxTempFanStart")             { maxTempFanStart = pub.payload_string().toInt(); }// проверяем из нужного
ли нам топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/deltaT")                    { deltaT = pub.payload_string().toInt(); }// проверяем из нужного ли нам топика
пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/deltaHumiditi")              { deltaHumiditi = pub.payload_string().toInt(); }// проверяем из нужного ли нам
топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/NumBankSave")                { NumBankSave = pub.payload_string().toInt(); }// проверяем из нужного ли
нам топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/IfBankSave")                 { IfBankSave = pub.payload_string().toInt(); }// проверяем из нужного ли нам
топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/BankSave")                   { BankSave = pub.payload_string().toInt(); }// проверяем из нужного ли нам
топика пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/ConstP")                    { BankSave = pub.payload_string().toInt(); }// проверяем из нужного ли нам топика
пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/ConstI")                    { BankSave = pub.payload_string().toInt(); }// проверяем из нужного ли нам топика
пришли данные // преобразуем полученные данные в тип integer
  if(String(pub.topic()) == "Hothouse/ConstD")                    { BankSave = pub.payload_string().toInt(); }// проверяем из нужного ли нам топика
пришли данные // преобразуем полученные данные в тип integer

}

void jsonGet() {

  if (!wclient.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
```

```
  Serial.println(wapi);
  Serial.println("Host: api.openweathermap.org");
  Serial.println("Connection: close");
  Serial.println();

 delay(1500);
 // Read all the lines of the reply from server and print them to Serial
 while(wclient.available()){
  line = wclient.readStringUntil('\r');
 }
 Serial.print(line);
 Serial.println();
 Serial.println("closing connection");
}

void WiFi_funk()
{
 voltage = ads.readADC_SingleEnded(1) * 0.185 / 1000 * 5.75;     //подсчет заряда аккумулятора
 if( voltage > 13.5 ) netpower = 1;
 else netpower = 0;
 delay(10);
 int i = 0;
  while (i<10){
  lcd.setCursor(18, 3); lcd.print("\8");          //Отправка значений на сервер
  delay(100);i++;
  }
  while (!client.connect(MQTT::Connect(mqtt_id).set_auth(mqtt_user, mqtt_pass))) {
  Serial.print(".");
  delay(1000);
 }
```

```
  Serial.println("Connected to MQTT server");

   lcd.setCursor(18, 3); lcd.print("\8");          //Отправка значений на сервер
     Serial.println("Start");delay(50);
  client.publish("Hothouse/Temp",String(Tnow));          Serial.println("Tnow = ");       Serial.println(Tnow);delay(50);
  client.publish("Hothouse/ControlTemp",String(Temp));    Serial.println("ControlTemp = "); Serial.println(Temp);delay(50);
  client.publish("Hothouse/deltaT",String(deltaT));       Serial.println("deltaT = "); Serial.println(deltaT);  delay(50);
  client.publish("Hothouse/Hum",String(hum));             Serial.println("HumiditiNow = "); Serial.println(hum);delay(50);
  client.publish("Hothouse/ControlHum",String(Humiditi)); Serial.println("ControlHum = ");  Serial.println(Humiditi);delay(50);
  client.publish("Hothouse/deltaHumiditi",String(deltaHumiditi));    Serial.println("deltaHumiditi = "); Serial.println(deltaHumiditi);delay(50);
  client.publish("Hothouse/Power",String(voltage));       Serial.println("Voltage = ");     Serial.println(voltage);delay(50);
  client.publish("Hothouse/NetPower",String(netpower));   Serial.println("NetPower = ");    Serial.println(netpower);delay(50);
  client.publish("Hothouse/ConstP",String(consKp));       Serial.println("ConstP = "); Serial.println(consKp);delay(50);
  client.publish("Hothouse/ConstI",String(consKi));       Serial.println("ConstI = "); Serial.println(consKi);delay(50);
  client.publish("Hothouse/ConstD",String(consKd));      Serial.println("ConstD = "); Serial.println(consKd);delay(250);
  client.subscribe("Hothouse/NumBankSave");              client.set_callback(callback);// подписывааемся по топик с данными для банка загрузки

     Serial.println("Finish");
     Serial.println("");

   if(IfBankSave == 1){
   IfBankSave = 0;
   while(BankSave<1){
   if(BankSave == 1){SaveToEEPROM(NumBankSave);break;}
   Serial.println("");
  client.subscribe("Hothouse/deltaHumiditi");client.set_callback(callback); // подписывааемся по топик с данными для светодиода
  client.subscribe("Hothouse/deltaT"); client.set_callback(callback);// подписывааемся по топик с данными для дельты температуры
  client.subscribe("Hothouse/maxTempFanStart"); client.set_callback(callback);// подписывааемся по топик с данными для критичной
температуры
  client.subscribe("Hothouse/TimeFanWork"); client.set_callback(callback);// подписывааемся по топик с данными для работы вентилятора
```

```
  client.subscribe("Hothouse/TimeIntervalFanWork "); client.set_callback(callback);// подписываaемся по топик с данными для периода
вентилятора
  client.subscribe("Hothouse/ControlHum"); client.set_callback(callback);// подписываaемся по топик с данными для заданной температуры
  client.subscribe("Hothouse/ControlTemp"); client.set_callback(callback);// подписываaемся по топик с данными для заданной влажности
  client.subscribe("Hothouse/BankSave"); client.set_callback(callback);// подписываaемся по топик с данными для банка загрузки
  client.subscribe("Hothouse/IfBankSave"); client.set_callback(callback);// подписываaемся по топик с данными для подтверждения
перезагрузки
  client.subscribe("Hothouse/ConstP"); client.set_callback(callback);// подписываaемся по топик с данными для подтверждения перезагрузки
  client.subscribe("Hothouse/ConstI"); client.set_callback(callback);// подписываaемся по топик с данными для подтверждения перезагрузки
  client.subscribe("Hothouse/ConstD"); client.set_callback(callback);// подписываaемся по топик с данными для подтверждения перезагрузки
    Serial.println("");



    }
    }
jsonGet();
     StaticJsonBuffer<2000> jsonBuffer;                  /// буфер на 2000 символов
   JsonObject& root = jsonBuffer.parseObject(line);     // Обрабатываем String
   if (!root.success()) {
    Serial.println("parseObject() failed");             // если ошибка, сообщаем об этом
     jsonGet();                                         // опрашиваем сервер еще раз
    return;                                            // и запускаем заново
  }

  tempK = root["main"]["temp"];                 // достаем температуру из структуры main
  tempC = tempK - 273.15;                       // переводим кельвины в цельси
  tempKmin = root["main"]["temp_min"];          // и так далее
  tempCmin = tempKmin - 273.15;
  tempKmax = root["main"]["temp_max"];
  tempCmax = tempKmax - 273.15;
```

```
  pressurehPa = root["main"]["pressure"];         // достаем давление из структуры main
  pressure = pressurehPa/1.333;
  humidity = root["main"]["humidity"];            // достаем влажность из структуры main
  windspeed = root["wind"]["speed"];             // достаем скорость ветра из структуры main
  winddeg = root["wind"]["deg"];                 // достаем направление ветра из структуры main

}



void loop()
{

Temp=20;consKp=1;consKi=1;consKd=1;HumGround=80;deltaT=0.1;
TypeHouse=1;Humiditi=80;deltaHumiditi=1;maxTempFanStart=30;TimeFanWork=120;TimeIntervalFanWork=300;FanWorkFlag=0;

  PressingButtons = 0;
  currentMillis = millis();
   RTC.getTime();

  if (FlagMenu == 0)
  {
    NOWyear = RTC.year; NOWmonth = RTC.month; NOWday = RTC.day; NOWhour = RTC.hour; NOWminute = RTC.minute; NOWsecond =
RTC.second;
  }
  if (currentTime >= (loopTime))
  {
    PressKeyMenu();
    switch (PressingButtons) {
case 1: {                                                    // обработка события нажатия кнопки "МЕНЮ"
```

```
    if ((MainMenu == 0 && SubMenu == 0) || (MainMenu == 0 && SubMenu == 1)) { MainMenu = 1;  SubMenu = 0;  StartMillis = currentMillis;
delay(200); }
    else {
     if (MainMenu == 4 && SubMenu == 0) { PreSetTime(); SubMenu = 1; StartMillis = currentMillis;  delay(200); }
     else {
      if (MainMenu != 0 && SubMenu == 0) { SubMenu = 1;  StartMillis = currentMillis;  delay(200); }
      else {
       if (MainMenu == 4 && SubMenu != 0) { SetTime(); SubMenu = 0; FlagMenu = 0; StartMillis = currentMillis; delay(200); }
       else { if (MainMenu != 0 && SubMenu != 0) { SubMenu = 0;  FlagMenu = 0;  StartMillis = currentMillis;  delay(200); } }
      }
     }
    }break; }
   case 2: {                                                                    // обработка события нажатия кнопки "ВВЕРХ"
    if (MainMenu != 0 && SubMenu == 0) { MainMenu--;  StartMillis = currentMillis;  delay(200);  if (MainMenu < 1)  MainMenu = 6; }
    else {
     if (MainMenu == 1 && SubMenu != 0 && FlagMenu == 0) { SubMenu--;  StartMillis = currentMillis;  delay(200);  if (SubMenu < 1)
SubMenu = 9; }
     else {
      if (MainMenu == 4 && SubMenu != 0 && FlagMenu == 0) { SubMenu--;  StartMillis = currentMillis;  delay(200);  if (SubMenu < 1)
SubMenu = 6; }
      else {
       if (MainMenu == 5 && SubMenu != 0 && FlagMenu == 0) { SubMenu--;  StartMillis = currentMillis;  delay(200);  if (SubMenu < 1)
SubMenu = 5; }
       else {
        if (MainMenu == 6 && SubMenu != 0 && FlagMenu == 0) { SubMenu--;  StartMillis = currentMillis;  delay(200);  if (SubMenu < 1)
SubMenu = 3; }
        else {
         if (MainMenu == 1 && FlagMenu == 1) {
          switch (SubMenu) {
          case 1: {  Temp += FadeAmountTemp;          StartMillis = currentMillis; if (Temp >= maxTempDanger)Temp = maxTempDanger;
```

```
delay(100);  break; }
            case 2: {  deltaT += FadeAmountTemp;         StartMillis = currentMillis; if (deltaT >= maxdeltaT)deltaT = maxdeltaT;       delay(100);
break; }
            case 3: {  Humiditi += FadeAmountHum;        StartMillis = currentMillis; if (Humiditi >= MaximumHumiditi)Humiditi =
MaximumHumiditi;  delay(100);  break; }
            case 4: {  k++; if (k > 3) k = 3; TypeHouse = VarHouse[k]; StartMillis = currentMillis; delay(200); EEPROM.write(135, TypeHouse);
delay(100); break; }
            case 5: {  deltaHumiditi += + FadeAmountdeltaHum; StartMillis = currentMillis; if (deltaHumiditi >= maxdeltaHum)deltaHumiditi =
maxdeltaHum;          delay(100);  break; }
            case 6: {  StartMillis = currentMillis; if (FanWorkFlag == 0) FanWorkFlag = 1; else if (FanWorkFlag == 1) FanWorkFlag = 0;
delay(100);break; }
           }
          }
          else {
           if (MainMenu == 2 && FlagMenu == 0 && SubMenu == 1) { bank++; StartMillis = currentMillis;  delay(200); if (bank > 4) bank = 4;
}
           else {
            if (MainMenu == 3 && FlagMenu == 0 && SubMenu == 1) { bank++; StartMillis = currentMillis;  delay(200); if (bank > 4) bank = 4;
}
            else {
             if (MainMenu == 4 && FlagMenu == 1) {
              switch (SubMenu) {
              case 1: {  NOWyear++;    if (NOWyear > 2030)  NOWyear = 2030;  delay(200);  GoSetTime(NOWyear);  break; }
              case 2: {  NOWmonth++;   if (NOWmonth > 12)   NOWmonth = 12;   delay(200);  GoSetTime(NOWmonth);  break; }
              case 3: {  NOWday++;     if (NOWday > 31)     NOWday = 31;     delay(200);  GoSetTime(NOWday);  break; }
              case 4: {  NOWhour++;    if (NOWhour > 23)    NOWhour = 23;    delay(200);  GoSetTime(NOWhour);  break; }
              case 5: {  NOWminute++;  if (NOWminute > 59)  NOWminute = 59;  delay(200);  GoSetTime(NOWminute);  break; }
              case 6: {  NOWsecond++;  if (NOWsecond > 59)  NOWsecond = 59;  delay(200);  GoSetTime(NOWsecond);  break; }
              }
             }
```

```
              else {
                if (MainMenu == 5 && FlagMenu == 1) {
                  switch (SubMenu) {
                  case 1: {  consKp += 0.01;                StartMillis = currentMillis; if (consKp >= 100)consKp = 100;      delay(100);
EEPROM.write(131,consKp);  break; }
                  case 2: {  consKi += 0.01;                StartMillis = currentMillis; if (consKp >= 100)consKp = 100;      delay(100);
EEPROM.write(132,consKi);  break; }
                  case 3: {  consKd += 0.01;                StartMillis = currentMillis; if (consKp >= 100)consKp = 100;      delay(100);
EEPROM.write(133,consKd);  break; }
                  case 4: { maxTempFanStart += FadeAmountTemp;                StartMillis = currentMillis; if (maxTempFanStart >=
maxTempFanStartMenuMax)maxTempFanStart = maxTempFanStartMenuMax;        delay(100);  break; }
                  case 5: { TimeFanWork += FadeTimeFanWork; StartMillis = currentMillis; if (TimeFanWork >= maxTimeFanWork)
TimeFanWork = maxTimeFanWork; TimeFaningInterval = TimeFanWork * 1000;  delay(100);  break; }
                  case 6: { TimeIntervalFanWork += FadeTimeIntervalFanWork; StartMillis = currentMillis; if (TimeIntervalFanWork >=
maxTimeIntervalFanWork)TimeIntervalFanWork = maxTimeIntervalFanWork; TimeIntervalFaningInterval = TimeIntervalFanWork * 60000;
delay(100);  break; }
                }
                   }
                  }
                 }
                }
               }
              }
             }
            }
           }
          } break;
         }
        case 3: {                                                    // обработка события нажатия кнопки "ВНИЗ"
          if (MainMenu != 0 && SubMenu == 0) { MainMenu++;  StartMillis = currentMillis;  delay(200);  if (MainMenu > 6)  MainMenu = 1; }
```

```
    else {
     if (MainMenu == 1 && SubMenu != 0 && FlagMenu == 0) { SubMenu++;  StartMillis = currentMillis;  delay(200);  if (SubMenu > 9)
SubMenu = 1; }
      else {
       if (MainMenu == 4 && SubMenu != 0 && FlagMenu == 0) { SubMenu++;  StartMillis = currentMillis;  delay(200);  if (SubMenu > 6)
SubMenu = 1; }
       else {
        if (MainMenu == 5 && SubMenu != 0 && FlagMenu == 0) { SubMenu++;  StartMillis = currentMillis;  delay(200);  if (SubMenu > 6)
SubMenu = 1; }
        else {
         if (MainMenu == 6 && SubMenu != 0 && FlagMenu == 0) { SubMenu++;  StartMillis = currentMillis;  delay(200);  if (SubMenu > 3)
SubMenu = 1; }
         else {
          if (MainMenu == 1 && FlagMenu == 1) {
           switch (SubMenu) {
           case 1: {  Temp -= FadeAmountTemp;                StartMillis = currentMillis;  if (Temp <= minTempDanger)  Temp =
minTempDanger;      delay(100);  break; }
           case 2: {  deltaT -= FadeAmountTemp;              StartMillis = currentMillis;  if (deltaT <= mindeltaT)  deltaT = mindeltaT;
delay(100);  break; }
           case 3: {  Humiditi -= FadeAmountHum;             StartMillis = currentMillis;  if (Humiditi <= MinimumHumiditi)  Humiditi =
MinimumHumiditi;  delay(100);  break; }
           case 4: {  deltaHumiditi -= FadeAmountdeltaHum;  StartMillis = currentMillis;  if (deltaHumiditi <= mindeltaHum)  deltaHumiditi =
mindeltaHum;          delay(100);  break; }
           case 5: {  k--; if (k < 0) k = 0; TypeHouse = VarHouse[k]; StartMillis = currentMillis; delay(200); EEPROM.write(135, TypeHouse);
delay(100); break; }
           case 6: { StartMillis = currentMillis; if (FanWorkFlag == 0) FanWorkFlag = 1; else if (FanWorkFlag == 1) FanWorkFlag = 0;
delay(100);break; }
          }
          }
          else {
```

```
            if (MainMenu == 2 && FlagMenu == 0 && SubMenu == 1) { bank--; StartMillis = currentMillis;  delay(200); if (bank < 0) bank = 0; }
            else {
              if (MainMenu == 3 && FlagMenu == 0 && SubMenu == 1) { bank--; StartMillis = currentMillis;  delay(200); if (bank < 0) bank = 0;
}
              else {
                if (MainMenu == 4 && FlagMenu == 1) {
                  switch (SubMenu) {
                  case 1: {  NOWyear--;    StartMillis = currentMillis;  if (NOWyear < 2015) NOWyear = 2015; delay(200);  GoSetTime(NOWyear);
break; }
                  case 2: {  NOWmonth--;   StartMillis = currentMillis;  if (NOWmonth < 1) NOWmonth = 1;   delay(200);  GoSetTime(NOWmonth);
break; }
                  case 3: {  NOWday--;     StartMillis = currentMillis;  if (NOWday < 1) NOWday = 1;     delay(200);  GoSetTime(NOWday);  break;
}
                  case 4: {  NOWhour--;    StartMillis = currentMillis;  if (NOWhour < 0) NOWhour = 0;    delay(200);  GoSetTime(NOWhour);
break; }
                  case 5: {  NOWminute--;  StartMillis = currentMillis;  if (NOWminute < 0) NOWminute = 0;  delay(200);
GoSetTime(NOWminute);  break; }
                  case 6: {  NOWsecond--;  StartMillis = currentMillis;  if (NOWsecond < 0) NOWsecond = 0;  delay(200);
GoSetTime(NOWsecond);  break; }
                   }
                 }
                 else {
                  if (MainMenu == 5 && FlagMenu == 1) {
                    switch (SubMenu) {
                    case 1: {  consKp -= 0.01;               StartMillis = currentMillis; if (consKp >= 0.01)consKp = 0.01;      delay(100);
EEPROM.write(131,consKp); break; }
                    case 2: {  consKi -= 0.01;               StartMillis = currentMillis; if (consKp >= 0.01)consKp = 0.01;      delay(100);
EEPROM.write(132,consKi); break; }
                    case 3: {  consKd -= 0.01;               StartMillis = currentMillis; if (consKp >= 0.01)consKp = 0.01;      delay(100);
EEPROM.write(133,consKd); break; }
```

```
                case 4: { maxTempFanStart -= FadeAmountTemp;                    StartMillis = currentMillis;  if (maxTempFanStart <=
maxTempFanStartMenuMin)  maxTempFanStart = maxTempFanStartMenuMin;        delay(100);  break; }
                case 5: { TimeIntervalFanWork -= FadeTimeIntervalFanWork;     StartMillis = currentMillis;  if (TimeIntervalFanWork <=
minTimeIntervalFanWork)  TimeIntervalFanWork = minTimeIntervalFanWork;  TimeIntervalFaningInterval = TimeIntervalFanWork * 60000;
delay(100);  break; }
                case 6: { TimeFanWork -=FadeTimeFanWork;                    StartMillis = currentMillis;  if (TimeFanWork <=
minTimeFanWork)  TimeFanWork = minTimeFanWork; TimeFaningInterval = TimeFanWork * 1000; delay(100);  break; }
              }
                  }
                 }
                }
               }
              }
             }
            }
           }
          }
         }
        }
       } break;
      }
     case 4: {                                                                  // обработка события нажатия кнопки "ВЫБОР"
      if (MainMenu == 0 && SubMenu == 0) { MainMenu = 0;  SubMenu = 1;  delay(200); }
      else {
       if (MainMenu == 0 && SubMenu == 1) { MainMenu = 0;  SubMenu = 0;  delay(200); }
       //else {  if ( MainMenu == 2 && SubMenu == 1 ) { SaveToEEPROM ( bank ); lcd.setCursor(0, 1); lcd.print( "    saving...   " ); }
       //else {  if ( MainMenu == 3 && SubMenu == 1 ) { LoadFromEEPROM ( bank ); lcd.setCursor(0, 1); lcd.print( "    loading...  " ); }
       else {
        if (MainMenu != 0 && SubMenu != 0 && FlagMenu == 0) { FlagMenu = 1;  delay(200); }
        else { if (MainMenu != 0 && SubMenu != 0 && FlagMenu == 1) { FlagMenu = 0;  delay(200); } }
       }
      }
```

```
   break; }
  case 5: {                                                    // обработка события нажатия кнопки "Открытие окон"
    StartFan();
    break; }
  }
}


  switch (MainMenu) {
  case 0:      // главное меню "0"
    switch (SubMenu) {
    case 0:  m = 0; break;  // вывод на экран главного меню "0" подменю "0"
    case 1:  m = 1; break;  // вывод на экран главного меню "0" подменю "1"
    }
    break;
  case 1:      // главное меню "1"
    switch (SubMenu) {
    case 0:  m = 10; break;  // вывод на экран главного меню "1" подменю "0"
    case 1:  m = 11; break;  // вывод на экран главного меню "1" подменю "1"
    case 2:  m = 12; break;  // вывод на экран главного меню "1" подменю "2"
    case 3:  m = 13; break;  // вывод на экран главного меню "1" подменю "3"
    case 4:  m = 14; break;  // вывод на экран главного меню "1" подменю "4"
    case 5:  m = 15; break;  // вывод на экран главного меню "1" подменю "5
    case 6:  m = 16; break;  // вывод на экран главного меню "1" подменю "5"
    }
    break;
  case 2:      // главное меню "2"
    switch (SubMenu) {
    case 0:  m = 20; break;  // вывод на экран главного меню "2" подменю "0"
    case 1:  m = 21; break; // вывод на экран главного меню "2" подменю "1"
```

```
  }
  break;
case 3:      // главное меню "3"
  switch (SubMenu) {
  case 0:  m = 30; break;  // вывод на экран главного меню "3" подменю "0"
  case 1:  m = 31; break;  // вывод на экран главного меню "3" подменю "1"
  }
  break;
case 4:
  switch (SubMenu) {
  case 0:  m = 40; break;  // вывод на экран главного меню "4" подменю "0"
  case 1:  m = 41; break;  // вывод на экран главного меню "4" подменю "1"
  case 2:  m = 42; break;  // вывод на экран главного меню "4" подменю "2"
  case 3:  m = 43; break;  // вывод на экран главного меню "4" подменю "3"
  case 4:  m = 44; break;  // вывод на экран главного меню "4" подменю "4"
  case 5:  m = 45; break;  // вывод на экран главного меню "4" подменю "5"
  case 6:  m = 46; break;  // вывод на экран главного меню "4" подменю "6"
  }
  break;
case 5:
  switch (SubMenu) {
  case 0:  m = 50; break;  // вывод на экран главного меню "5" подменю "0"
  case 1:  m = 51; break;  // вывод на экран главного меню "5" подменю "1"
  case 2:  m = 52; break;  // вывод на экран главного меню "5" подменю "2"
  case 3:  m = 53; break;  // вывод на экран главного меню "5" подменю "3"
  case 4:  m = 54; break;  // вывод на экран главного меню "5" подменю "4"
  case 5:  m = 55; break;  // вывод на экран главного меню "5" подменю "5"
  case 6:  m = 56; break;  // вывод на экран главного меню "5" подменю "6"
  }
  break;
```

```
   case 6:
     switch (SubMenu) {
     case 0:  m = 60; break;  // вывод на экран главного меню "6" подменю "0"
     case 1:  m = 61; break;  // вывод на экран главного меню "6" подменю "1"
     case 2:  m = 62; break;  // вывод на экран главного меню "6" подменю "2"
     case 3:  m = 63; break;  // вывод на экран главного меню "6" подменю "3"
     }
   }
```

```
  switch (m) {
  case 0: {  lcd.clear();lcd.setCursor(0, 0);  lcd.print("T="); lcd.print(Tnow); lcd.print("\3 (");lcd.print(Temp);    lcd.print("\3)");
           lcd.setCursor(0, 1);  lcd.print("H="); lcd.print(hum);  lcd.print("%("); lcd.print(Humiditi); lcd.print("%)");
           lcd.setCursor(0, 2);if( voltage >= 13.5 ){lcd.print("220V - ON"); }else{ lcd.print("Bat="); lcd.print(voltage);}
           lcd.setCursor(0, 3);
           if (RTC.hour < 10) lcd.print(0); lcd.print(RTC.hour); lcd.print(":"); if (RTC.minute < 10) lcd.print(0); lcd.print(RTC.minute); lcd.print(":");
if (RTC.second < 10) lcd.print(0); lcd.print(RTC.second);
            BME280Read(); StartFan();StartHot();StartLite();StartHum(); TimerCalculatePrint();WiFi_funk();
FlagMenu = 0;   break; }
  case 10: {  lcd.clear(); lcd.setCursor(0, 1); lcd.print(F("    Setting     ")); lcd.setCursor(0, 2); lcd.print(F("   Growing  ")); lcd.setCursor(15, 1);
lcd.print("\1               "); delay(100);FlagMenu = 0;   break; }
  case 11: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("   Temperature  ")); lcd.setCursor(0, 1); lcd.print(F("t =         ")); lcd.print(Temp);
lcd.print("\3               "); PrintMenuWrite(FlagMenu); delay(100); break; }
  case 12: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("    Delta T     ")); lcd.setCursor(0, 1); lcd.print(F("\4t =       ")); lcd.print(deltaT);
lcd.print("\3               "); PrintMenuWrite(FlagMenu); delay(100); break; }
  case 13: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("   Humiditi     ")); lcd.setCursor(0, 1); lcd.print(F("H =         ")); lcd.print(Humiditi);
lcd.print("%               "); PrintMenuWrite(FlagMenu); delay(100); break; }
  case 14: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("    Delta H     ")); lcd.setCursor(0, 1); lcd.print(F("\4h =       ")); lcd.print(deltaHumiditi);
```

```
lcd.print("%                ");          PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 15: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  Type HotHouse ")); lcd.setCursor(0, 1); lcd.print(F("Type =      ")); TypeHousePrint();
lcd.print("                ");          PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 20: {  lcd.clear(); lcd.setCursor(0, 1); lcd.print(F("  Save setting  ")); lcd.setCursor(0, 2); lcd.print(F("  to  EEPROM")); lcd.setCursor(15, 1);
lcd.print("\1               ");                                                  delay(100); FlagMenu = 0;  break; }
  case 21: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("Save setting to ")); lcd.setCursor(0, 1); lcd.print(F("bank        ")); lcd.print(bank); if
(FlagMenu == 0) {      lcd.print(" press set"); delay(100); } else { SaveToEEPROM(bank);   lcd.print(" saving..."");   delay(100); }              break; }
  case 30: {  lcd.clear(); lcd.setCursor(0, 1); lcd.print(F("  Load setting  ")); lcd.setCursor(0, 2); lcd.print(F("from  EEPROM")); lcd.setCursor(15, 1);
lcd.print("\1               ");                                                  delay(100); FlagMenu = 0;  break; }
  case 31: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  Load setting  ")); lcd.setCursor(0, 1); lcd.print(F("bank ")); lcd.print(bank); if (FlagMenu
== 0) {      lcd.print(" press set"); delay(100); } else { LoadFromEEPROM(bank); lcd.print(" loading..");   delay(100); }              break; }
  case 40: {  lcd.clear(); lcd.setCursor(0, 1); lcd.print(F("      Time      ")); lcd.setCursor(0, 2); lcd.print(F("  setting   ")); lcd.setCursor(15, 1);
lcd.print("\1               ");                                                  delay(100); FlagMenu = 0;  break; }
  case 41: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  setup  year   ")); lcd.setCursor(5, 1); lcd.print(Setyear);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 42: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  setup  month  ")); lcd.setCursor(6, 1); lcd.print(Setmonth);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 43: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("   setup  day   ")); lcd.setCursor(6, 1); lcd.print(Setday);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 44: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  setup  hour   ")); lcd.setCursor(6, 1); lcd.print(Sethour);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 45: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  setup minute  ")); lcd.setCursor(6, 1); lcd.print(Setminute);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 46: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("  setup second  ")); lcd.setCursor(6, 1); lcd.print(Setsecond);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 50: {  lcd.clear(); lcd.setCursor(0, 1); lcd.print(F("    Exstra      ")); lcd.setCursor(0, 2); lcd.print("     Setting  ");  lcd.setCursor(15, 1);
lcd.print("\1               ");                                                  delay(100); FlagMenu = 0;  break; }
  case 51: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("      K p       ")); lcd.setCursor(0, 1); lcd.print(F("P = "));           lcd.print(consKp);
PrintMenuWrite(FlagMenu);                delay(100);              break; }
  case 52: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("      K i       ")); lcd.setCursor(0, 1); lcd.print(F("I = "));           lcd.print(consKi);
```

```
PrintMenuWrite(FlagMenu);                delay(100);          break; }
  case 53: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("      K d       ")); lcd.setCursor(0, 1); lcd.print(F("D = "));          lcd.print(consKd);
PrintMenuWrite(FlagMenu);                delay(100);          break; }
  case 54: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("   Cooling T    ")); lcd.setCursor(0, 1); lcd.print(F("t = "));
lcd.print(maxTempFanStart);     lcd.print("\3              ");          PrintMenuWrite(FlagMenu);                delay(100);          break; }
  case 55: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("CO2 faning time ")); lcd.setCursor(0, 1); lcd.print(F("time = "));
lcd.print(TimeFanWork);         lcd.print(" sec.            ");          PrintMenuWrite(FlagMenu);                delay(100);          break; }
  case 56: {  lcd.clear(); lcd.setCursor(0, 0); lcd.print(F("CO2 fan interval")); lcd.setCursor(0, 1); lcd.print(F("time = "));
lcd.print(TimeIntervalFanWork); lcd.print(" min.            ");          PrintMenuWrite(FlagMenu);                delay(100);          break; }
  }
 }
```