Одеський національний університет імені І.І.Мечникова

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій

(повна назва кафедри (предметної, циклової комісії))

# Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

Розробка системи автоматичного поливу рослин з автономним живленням

(назва українською)

Development of automatic irrigation system with autonomous power supply

(назва англійською)

Виконав: студент денної форми навчання
спеціальності 123 Комп'ютерна інженерія.

(шифр і назва напряму підготовки, спеціальності)

Савіних Микола Олександрович

(прізвище, ім'я, по-батькові)

Керівник ст. викл. Берков Ю.М. _____

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент д.т.н., проф. Гунченко Ю.О. _____

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2022 р.

Завідувач кафедри

_____ _Ю.О. Гунченко

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «_ _» ____ 2022 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Н.Ф. Казакова_

(підпис) (прізвище, ініціали)

Одеса – 2022

# АНОТАЦІЯ

Метою даної є розробка системи автоматичного управління зрошенням ґрунту.

В результаті аналізу визначено, що планування періодичності та часу зрошення обумовлено складом ґрунту, кліматичними умовами та безліччю факторів, які важко піддаються точному розрахунку. Основним параметром, що визначає необхідність зрошення, є кількість доступної рослин вологи, а основним параметром, що характеризує роботу системи зрошення, є ефективність системи зрошення.

Для досягнення мети роботи виконано розробку основних компонентів системи автоматичного управління зрошенням ґрунту: датчика для визначення кількості доступної для рослин вологи, пристрою вимірювання параметрів, керуючого пристрою, пристрою збору даних та програмного забезпечення.

Система пройшла апробацію в польових умовах та показала свою працездатність, надійність та ефективність у виконанні поставленого завдання.

Система може бути використана сільськогосподарськими підприємствами та фермерськими господарствами для підвищення врожайності.

Ключові слова: зрощення, arduino, labview, lora.

# ABSTRACT

The purpose of this work is to develop an automatic control system for soil irrigation.

As a result of the analysis, it was determined that planning the frequency and time of irrigation is due to the composition of the soil, climatic conditions and many factors that are difficult to accurately calculate. The main parameter that determines the need for irrigation is the amount of moisture available to plants, and the main parameter that characterizes the operation of the irrigation system is the efficiency of the irrigation system.

To achieve the goal of the work, the development of the main components of the automatic control system for soil irrigation was carried out: a sensor for determining the amount of moisture available to plants, a device for measuring parameters, a control device, a data collection device and software.

The system was tested in the field and showed its operability, reliability and efficiency in performing the task.

The system can be used by agricultural enterprises and farms to increase yields.



Key words: irrigation, arduino, labview, lora.

**ЗМІСТ**

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

PR – швидкість подачі води в одиницю часу

AW – кількість доступної води

EP – польова ємність

PWP – точка в'янення рослин

ET – швидкість евапотранспірації

EA – Ефективність системи зрошення

FR – Періодичність зрошення

TR – Час роботи зрошення

ПК – персональний комп'ютер

МК – мікроконтролер

ПЗ – програмне забезпечення

САУ – система автоматичного керування

ПВП – пристрій вимірювання та передачі параметрів

ПКЗ – пристрій керування зрошенням

ШПД – шлюз передачі даних

UART – універсальний асинхронний приймач-передавач

# ВСТУП

Використання систем автоматичного поливу (САП) на фермерських господарствах та сільськогосподарських підприємствах дозволяє суттєво підвищити врожайність, скоротити витрати та підвищити рентабельність виробництва сільськогосподарської продукції.

Активне використання САП стримується порівняно високими витратами купівлю устаткування, над ринком пропонується переважно продукція зарубіжного виробництва, і навіть відсутністю електропостачання полів задля забезпечення його роботи.

Метою данної роботи є розробка проекту системи автоматичного поливу з автономним живленням на основі платформи Arduino.

Завданням роботи є огляд предметної області, формування функціональних вимог, підбір апаратних компонентів та створення програмного коду для керування системою. Електричне живлення системи має здійснюється від сонячних панелей. Управління САП та моніторинг параметрів має відбуватися віддалено через мережу інтернет.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Огляд систем зрошення

Вода забезпечує протікання всіх фізичних процесів планети Земля, як і атмосфері, і у навколишньому середовищі. Оптимальна вологість ґрунту для сільськогосподарських культур – запорука високого врожаю, оскільки рослини не можуть розвиватися, якщо вологи у ґрунті недостатньо. Крім того, вода виконує й інші функції у ґрунті:

1) вологість впливає на аерацію, концентрацію солей та токсичних речовин;
2) обумовлює структуру, пластичність та щільність;
3) регулює температуру та теплоємність;
4) запобігає вивітрюванню;
5) визначає час проведення польових работ.

Зрошення – комплекс робіт з поливу водою на земельних ділянках, спрямований на живлення ґрунту, покращення його властивостей та продуктивності, підвищення врожайності культур. Зрошення забезпечує кореневу систему рослин поживними речовинами, знижує температуру та збільшує вологість повітря. Зрошення – найкращий спосіб перемоги над посухою, який імітує природні процеси, зволожуючи ґрунт та повітря.

Продуктивність зрошуваних земель у 3-4 рази більша, ніж у орних угідь, де системи іригації відсутні. Навіть мінімальний приріст урожайності (10-15%) гарантує окупність штучних зрошувальних систем від 2-х до 4-х років.

На території України зрошується всього 8% орних земель, а штучно зрошувані землі, як правило, дають у 2-3 рази більше врожаю та його якість у рази краща, ніж там, де поливу немає. Конкурентні переваги будуть у всіх фермерських підприємств, які інвестуватимуть у штучне зрошення.

В умовах правильного, регулярного поливу виходить урожай високої якості, якщо ж рослини відчувають нестачу вологи – їх якість на порядок гірша. При правильному регулярному поливі знижується кількість болючих рослин, у рослин набагато сильніший імунітет і стійкість перед шкідниками.

Іригаційне обладнання – найкраще рішення, щоб застрахуватися від посухи, підвищити врожайність та прибуток, не збільшуючи площі полів (5).

Широкозахватна дощувальна машина – це технічні агрегати, за допомогою яких виробляється полив та зрошення фермерських угідь середньої та великої площі (від 50 га і більше).

Рисунок 1.1 – Широкозахватна дощувальна машина

Функціональні можливості та великий модельний ряд дозволяє застосовувати дощувальні машини, як на полях, що проектуються, так і на сформованих сільськогосподарських територіях. Це одна з найефективніших і найчастіше використовуваних технологій поливу, якій віддають перевагу близько 70% фермерських господарств, що вирощують зернові, овочеві, фруктові та декоративні культури. На думку експертів та багатьох фермерів це найкращий метод перемоги над посухою, покращення показників якості та

кількості врожаю, і відповідно збільшення прибутку. Їм віддають перевагу практично всі сучасні європейські фермерства, що вирощують зернові, технічні, декоративні культури, овочі та фрукти. Відбувається підживлення верхнього та нижнього шару ґрунту, кореневої системи рослин, підвищується вологість повітря.

Забір води здійснюється з різних джерел – водоканал, свердловина, накопичувальний бак тощо. Зрошувати рослини можна з використанням фільтрів від важких металів та домішок, а також з додаванням добрив та різних біостимуляторів.

Широкозахватні дощувальні установки на 20 – 80% економічніші і вигідніші, ніж будь-які інші системи поливу, наприклад краплинний полив. Вони дають можливість відстежувати ефективність, контролювати та регулювати процес зрошення, працюючи в різних режимах, у тому числі при малому тиску та натиску води, що суттєво економить енергоресурси. Швидка окупність техніки протягом 2-5 років завдяки підвищенню врожайності.

Кругова самодосувна дощувальна машина дозволяє отримати високі та стійкі врожаї зернових, овочевих та технічних культур, а також багаторічних трав, забезпечуючи їх якісний полив.

Рисунок 1.2 – Кругова самодосувна дощувальна машина.

Машина може працювати на ділянках зі складним рельєфом, що мають неглибокий родючий шар, що виключає дороге планування зрошуваних полів. Високий рівень рівномірності розподілу дощу дає можливість одночасно з поливом проводити внесення рідких і розчинних мінеральних добрив. 3-4 дощувальні машини може обслуговувати лише один оператор, що підвищує економічність їхньої експлуатації. Далекоструминні кінцеві апарати дозволяють поливати поля прямокутної форми, автоматично включаючись на цій ділянці.

Завод «Фрегат» розробив принципово нову модель ферменної дощувальної машини із електричним приводом типу ДМФ «Фрегат» (6). Нова модель дощувальної машини випускається з 2010 року і дозволяє досягти максимальної економії електроенергії на подачу води завдяки низькому тиску на вході в машину. ДМФ «Фрегат» має можливість як кругового, і фронтального руху, і навіть фронтального руху з круговим переміщенням (іподромного типу). Основні технічні характеристики ДМФ «Фрегат» та якість поливу відповідають рівню кращих зарубіжних аналогів за меншої ціни. У дощувальній машині ДМФ «Фрегат» застосовані комплектуючі передових світових фірм – колісні та мотор-редуктори UMC (США), низьковисючі дощувальні апарати i-Wob Senninger (США), автоматична система управління з використанням комплектуючих Schneider-Electric та Honeywell. Машина оснащується електроприводом, кожен візок спирається на два пневмо колеса, що приводяться в рух власним мотор-редуктором за допомогою двох черв'ячних редукторів. Автоматична система управління забезпечує можливість прямого та реверсивного руху в широкому діапазоні робочих швидкостей.

Дощовий барабан – це система поливу, що адаптується до різних висот, швидкостей переміщення, полів незвичайної форми. Використовуються вони для зрошення всіх видів сільськогосподарських культур за рахунок своєї компактності та можливості швидкого транспортування. До того ж вартість

дощувальних барабанів дешевша, ніж у фронтальних або кругових широкозахватних машин (5).

Рисунок 1.3 – Дощувальний барабан

Дощувальні установки барабанного типу можуть використовуватись як на землях агрохолдингів, так і у фермерських господарствах та невеликих сільськогосподарських підприємствах.

Мобільні дощувальні установки барабанного типу широко використовуються для зрошення всіх видів сільськогосподарських культур за рахунок своєї компактності та можливості швидкого транспортування. Це система поливу, що адаптується до різних висот, швидкостей переміщення, полів незвичайної форми. До того ж вартість дощувальних барабанів дешевша, ніж у фронтальних або кругових широкозахватних машин, але в деяких випадках вони можуть бути не менш ефективними.

Багато фермерів вибирають саме барабанні дощувальні установки, оскільки вони мають низку істотних переваг:

1) Компактні та легко транспортуються з одного поля на інше;

2) Можна здійснювати полив кількох полів або ділянок однієї барабанної установки завдяки легкості її транспортування;
3) Дозволяють вносити рідкі добрива на поля;
4) Рослини можна зрошувати з використанням фільтрів від важких металів та домішок, а також з додаванням добрив та різних біостимуляторів;
5) Вирівнювання рельєфу або прокладання підземного трубопроводу не є обов'язковим для встановлення дощувальних машин барабанного типу;
6) Можна здійснювати полив полів незалежно від рельєфу місцевості та перепаду висот, зручні для полів будь-якої форми та конфігурації.

Представлені вище системи зрошення застосовують у сільське господарство.

Для зрошення невеликих площ фермерських господарств та присадибних ділянок використовується обладнання таких виробників як Hunter (7), Rain Bird (8), Irritroll (9):

Рисунок 1.4 – Розпилювачі роторного типу;

Рисунок 1.5 – Розпилювачі типу "спрей"

Рисунок 1.6 – Труби для крапельного поливу

Незалежно від типу та моделі, основним параметром, що характеризує систему зрошення, є швидкість подачі води в одиницю часу (PR) (7):

$$PR = \frac{Qw * 60}{S} \quad (1.1)$$

де:

Qw – витрата води, л/хв;

Sf – загальна площа зрошення, м2.

## 1.2 Вологість ґрунту

Природні ґрунти включають три основні фази: тверду, рідку та газоподібну (рис.1.4).

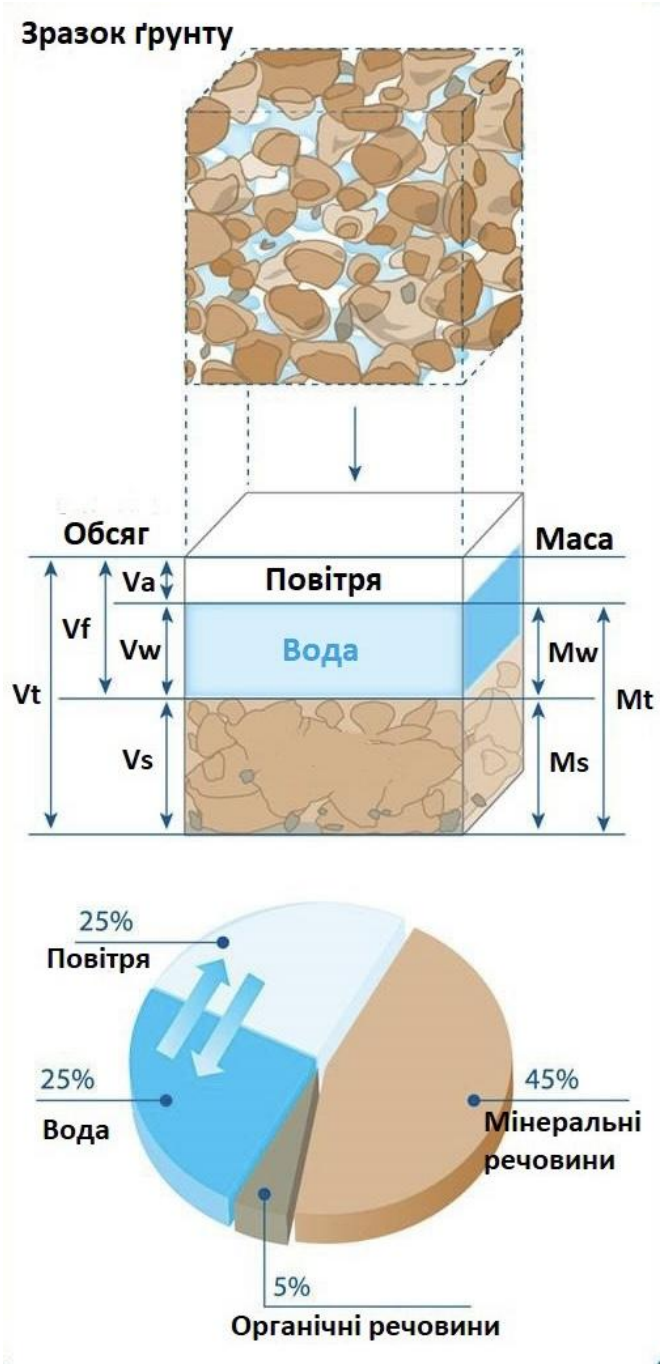



Рисунок 1.4 – Основні фазові складові ґрунту

Кожна з трьох фаз, у свою чергу, складається з домішок сполук.

Тверда фаза (мінерали та органічні сполуки) має складний склад та текстуру (10).

Діаграма найпоширеніших класів текстури ґрунту від піску до глини представлена на рисунку1.5.

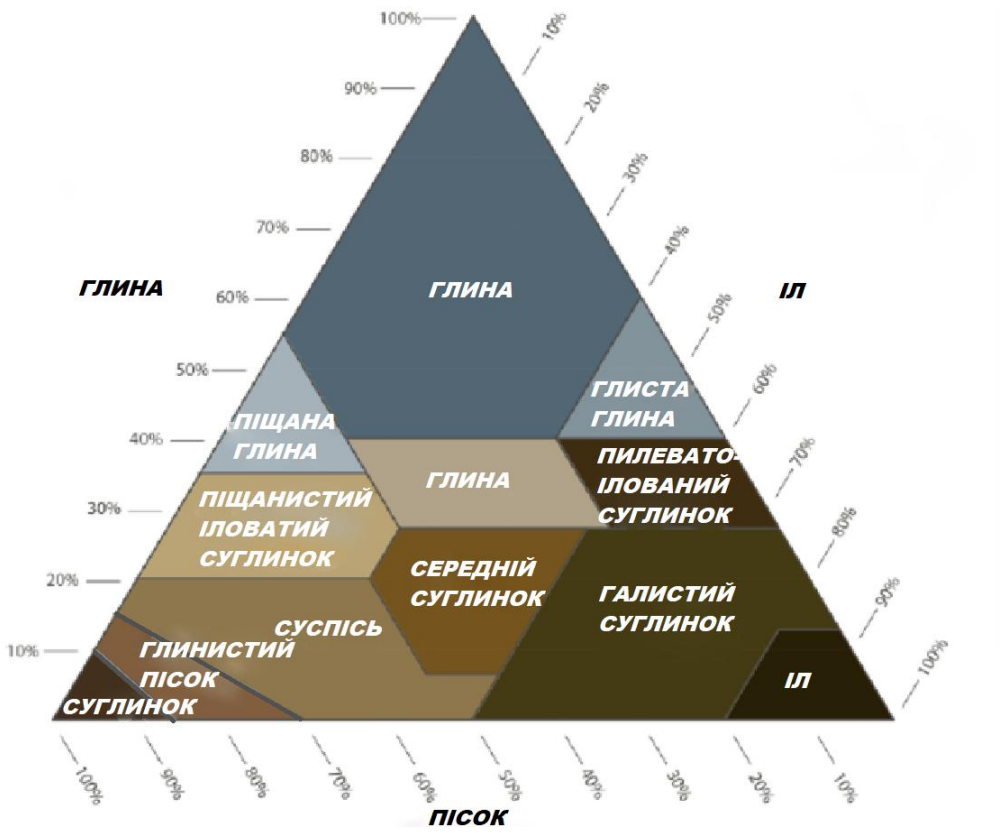



Рисунок 1.5 – Текстурний трикутник ґрунту

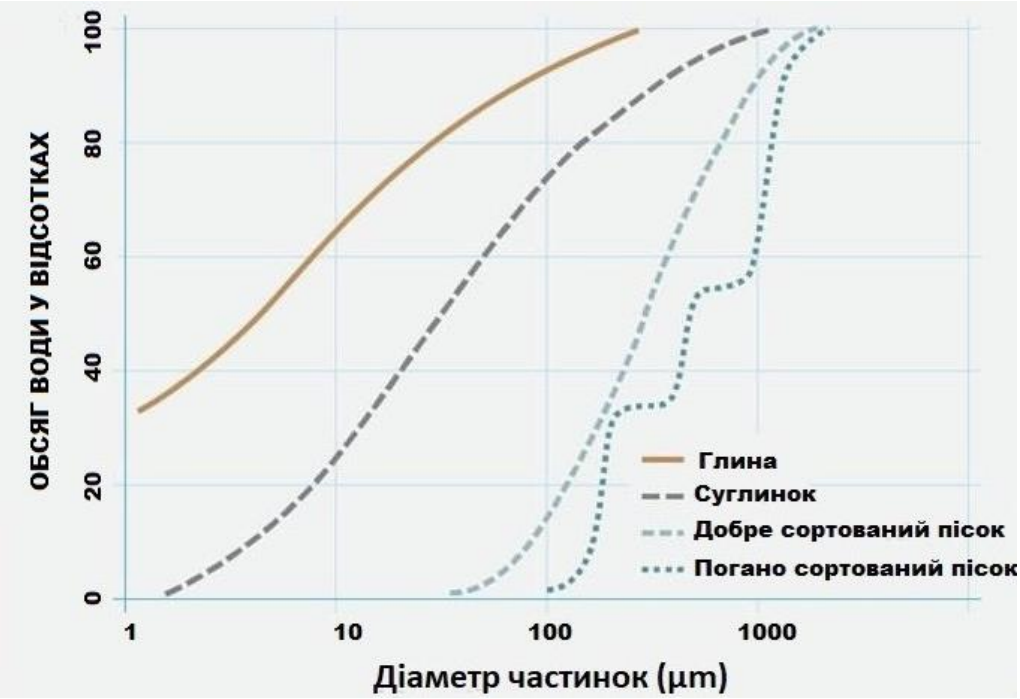



Рисунок 1.6 – Приклад розподілу частинок за розмірами, залежно від текстури ґрунту.

Крім текстури, до основних показників, що характеризують склад ґрунту, відноситься пористість.

Пористість Ns ґрунту визначається ставленням:

$$Ns = \frac{Vt - Vs}{Vt}, \quad (1.2)$$

де:

Vt – обсяг ґрунту;

Vs – обсяг твердої фази ґрунту.

Кількість води у ґрунті характеризується вологістю ґрунту. Вологість ґрунту Sm визначається співвідношенням маси води до маси твердої фази ґрунту та виражається у відсотковому співвідношенні:

$$Sm = \frac{100 * Mw}{Ms} \quad (1.3)$$

де:

Mw – маса води;

Ms – маса твердої фази ґрунту.

Об'ємний вміст води описує те саме, що й ваговий вміст води, за винятком того, що воно вказується на основі обсягу. З точки зору вмісту води, сухий ґрунт відповідає 0%, а 100% відповідає чистій воді. Таким чином, ґрунт ніколи не може мати 100% вологість з точки зору визначення за формулою (1.3).

Для визначення енергетичного стану води, що міститься у ґрунті, використовується водний потенціал. Це потенційна енергія води на одиницю відносного обсягу чистої води у нормальних умовах. Водний потенціал визначає тенденцію води переміщатися з однієї області в іншу через осмос, силу тяжіння, механічного тиску і матричних ефектів, таких як капілярний тиск. Концепція водного потенціалу є корисною для розуміння та розрахунку руху води всередині рослин та ґрунту. Водний потенціал зазвичай виявляється у вигляді потенційної енергії на одиницю об'єму:

$$\Psi = \Psi p + \Psi \pi + \Psi g + \Psi \mathrm{a} \quad (1.4)$$

де:

Ψp – потенціал тиску;

Ψπ – осмотичний потенціал;

Ψm – матричний потенціал;

Ψg – гравітаційний потенціал.

Потенціал тиску заснований на механічному тиску та є важливим компонентом загального водного потенціалу. Потенціал тиску збільшується, коли вода наповнює ґрунт.

Осмотичний потенціал залежить кількості розчинених у воді речовин. У чистої води осмотичний потенціал має велике значення. Осмотичний потенціал можливий завдяки присутності у ґрунті як неорганічних, так і органічних розчинених речовин. Оскільки молекули води дедалі більше злипаються навколо іонів чи молекул розчинених речовин, свобода руху, і,

отже, потенційна енергія води знижується. У міру збільшення концентрації розчинених речовин, осмотичний потенціал ґрунтового розчину знижується.

Коли вода знаходиться в контакті з твердими частинками (наприклад, глина або пісок частинки у ґрунті), адгезійні міжмолекулярні сили між водою та твердим тілом можуть бути значущими. Сили між молекулами води та твердими частинками у поєднанні з тяжінням між молекулами води сприяють поверхневому натягу та утворенню менісків усередині твердої матриці. Потім потрібна сила, щоб зламати ці меніски. Величина матричного потенціалу залежить від відстані між твердими частинками. У багатьох випадках абсолютне значення матричного потенціалу може бути відносно велике в порівнянні з іншими компонентами водного потенціалу. Він помітно знижує енергетичний стан води поблизу поверхонь частинок. Хоча рух води через матричний потенціал сповільнюється, він все ж таки надзвичайно важливий для подачі води до коріння рослин. Матричний потенціал завжди негативний, тому що вода, внаслідок сил адгезії, має нижчий енергетичний стан, ніж чиста вода. Матричний потенціал виникає лише у ненасиченому ґрунті над рівнем ґрунтових вод. Якщо матричний потенціал наближається до нуля, майже всі пори ґрунту повністю заповнені водою. Матричний потенціал може значно відрізнятися залежно від типу ґрунту.

Після рясного поливу вода заповнює весь простір між частинками ґрунту. У цьому значення гравітаційного потенціалу має максимальне значення. У міру того як вода дренує, гравітаційний потенціал зменшується до нуля за порівняно короткий час. При цьому матричний потенціал залишається найважливішим компонентом, тому що він пов'язаний з водою, яка знаходиться на поверхні частинок ґрунту. У міру того, як вода йде з ґрунту, простір, заповнений повітрям, стають більшими. А вода стає сильніше пов'язаною з частинками ґрунту.

Існує взаємозв'язок між водним потенціалом та об'ємним вмістом води, який можна проілюструвати за допомогою кривої утримання вологи у ґрунті.

На рисунку 1.7 показані зразкові криві для трьох різних ґрунтів. По осі абсцис відкладено водний потенціал у логарифмічному масштабі, а по осі ординат – об'ємний вміст води.

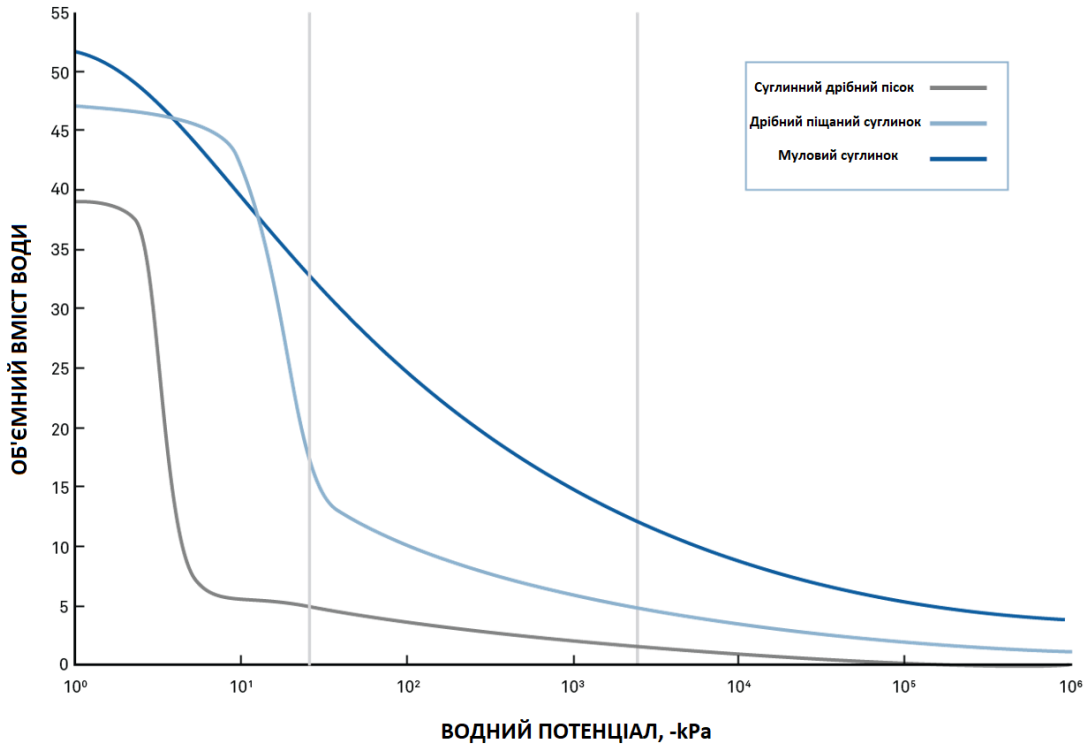



Рисунок 1.7 – Крива утримання вологи

Криві утримання вологи в ґрунті подібні до фізичних відбитків пальців, унікальних для кожного ґрунту. Це пов'язано з тим, що співвідношення між водним потенціалом і вмістом вологи в ґрунті є різним для кожного ґрунту. За допомогою цього співвідношення можна дізнатися, як різні ґрунти будуть вести себе у будь-якому місці кривої. З'являється можливість відповісти на такі важливі питання, як: чи буде вода швидко просочуватися через ґрунт або затримуватиметься в кореневій зоні? Криві утримання вологи у ґрунті – це інструмент, який використовується для прогнозування поглинання води рослинами та дренажу.

Доступна для рослин кількість води (AW) знаходиться між кількістю води, яка може утримуватися у ґрунті після інфільтрації та дренажу (польова

ємність ЕР) та точкою в'янення рослин (PWP), коли рослина вже не може витягувати з ґрунту воду.

У таблиці 1.1 та на рисунку 1.8 представлені деякі типові діапазони цих параметрів.

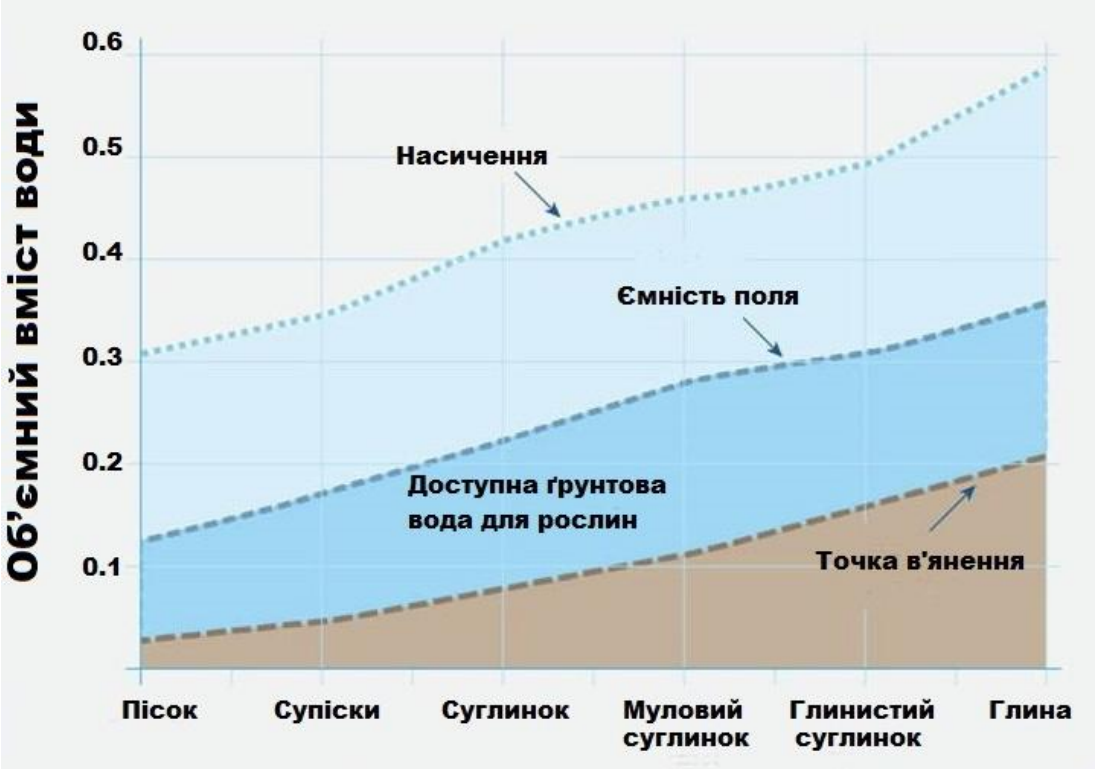



Рисунок 1.8. – Типові значення насичення, польової ємності та постійної точки в'янення для низки текстур ґрунту.

Таблиця 1.1 Типові значення параметрів ґрунту для різних текстур

| Текстура ґрунту | † Пористість, n(см3/см3) | * Насичена гідравлічна провідність, Ks(см/сек) | † Щільність, s | † Ємність поля (см3/см3) | † Постійна точка в'янення (см3/см3) |
|---|---|---|---|---|---|
| Піщаний | 0.38 | 10-3 – 10-1 | 1.65 | 0.09 | 0.04 |

Продовження таблиці 1.1

| | | | | | |
|---|---|---|---|---|---|
| Піщаний суглинок | 0.43 | 10-3 – 10-2 | 1.50 | 0.14 | 0.06 |
| Суглинок | 0.47 | 10-3 – 10-2 | 1.40 | 0.22 | 0.10 |
| Глинистий суглинок | 0.49 | 10-4 – 10-3 | 1.35 | 0.27 | 0.13 |
| Муліста глина | 0.51 | 10-7 – 10-4 | 1.30 | 0.31 | 0.15 |
| Глина | 0.53 | 0 – 10-4 | 1.25 | 0.35 | 0.17 |

Практичне значення концепцій польової ємності та точки в'янення полягає у визначенні діапазону КДВ. Для досягнення польової ємності після зволоження потрібен час (іноді від одного до кількох днів). Запас вологи у грунті, доступний рослинам, є важливим чинником щодо обсягу зрошення.

## 1.3 Планування зрошення

Далі буде розглянуто параметри, які необхідно враховувати під час управління зрошенням.

Першим, найважливішим параметром, є швидкість евапотранспірації ET (3). Параметр визначає кількість води, що втрачається внаслідок випаровування вологи з поверхні ґрунту та транспірації води рослиною. Приблизне значення евапотранспірації можна отримати з таблиці 1.2.

Таблиця 1.2 Середня швидкість евапотранспірації щодо різноманітних кліматичних умов.

| Таблиця ЄТ | |
|---|---|
| Клімат | мм/день |
| Холодний вологий | від 2.5 до 3.8 |
| Холодний сухий | від 3.8 до 5.1 |
| Теплий вологий | від 3.8 до 5.1 |
| Теплий сухий | від 5.1 до 6.3 |
| Гарячий вологий | від 5.1 до 7.6 |
| Гарячий сухий | від 7.6 до 11.4 |

Холодний – нижче 21oC.

Теплий – від 21 до 32oC.

Гарячий – вище 32oC.

Вологий – вище 50%.

Сухий – нижче 50%.

Різні рослини висувають особливі вимоги до поливу. Коефіцієнт урожаю Кс (3) дозволяє виразити цю зміну потреби у волозі. Таблиця 1.3 є посібником за коефіцієнтами врожаю для різних категорій звичайних рослин.

Таблиця 1.3 Коефіцієнт урожаю для різних рослин

| Коефіцієнт урожаю Кс для різних рослин | |
|---|---|
| Зрілі дерева | 0.80 |
| Чагарники (висотою понад 1 метр) | 0.70 |
| Кущі (висотою менше 1 метра) | 1.00 |
| Газон у теплий сезон | 0.50-0.70 |
| Газон у холодний сезон | 0.60-0.80 |

Глибокий і менш частий поливання рослин дає більше води в кореневу зону. Однак, щоб уникнути пошкодження рослин через те, що доступна вода падає до точки в'янення, необхідно запланувати час поливу доти, доки вся доступна вода буде використана.

Допустимий рівень виснаження (MAD) – цей параметр може змінюватись в залежності від типу ґрунту, ущільнення, глибини коренів та стресостійкості рослини.

Ефективність системи зрошення (EA) – це міра того, скільки води, що застосовується, доступно для використання в ефективній кореневій зоні. Це також показник того, наскільки добре система була спроектована та встановлена і наскільки добре вона керується. Параметр визначається шляхом розподілу кількості води у кореневій зоні на кількість води, використаної для зрошення. Хоча можна зробити досить точний розрахунок, але через вплив вітру, температури, вологості, типу ґрунту та глибини коренів, значення EA залишається обґрунтованою оцінкою. Для загальних цілей може розраховувати досягнення значення EA в межах від 60% до 80%.

Періодичність зрошення (у днях) визначається за такою формулою:

$$FR = \frac{AW * RZ * MAD}{ET \times Kc} \quad (1.4)$$

де:

AW = доступна вода;

RZ = коренева зона;

MAD = допустимий рівень виснаження;

ET = коефіцієнт евапотранспірації;

Kc = коефіцієнт урожаю.

Час роботи зрошення (у хвилинах) визначається за такою формулою:

$$TR = \frac{60 * FR * ET * Kc}{PR * EA} \quad (1.5)$$

де:

ET = швидкість евапотранспірації;

Kc = коефіцієнт урожаю;

PR = швидкість подачі води;

EA = ефективність системи.

Незалежно від того, який метод використовується для визначення часу роботи, він, як і раніше, є лише обґрунтованою оцінкою потреби рослин у воді. Належне керування зрошенням потребує спостереження за системою та рослинами, а також скорочення часу поливу для максимальної економії води.

Правильна техніка поливу важлива у розвиток глибокої кореневої системи. Поповнення доступної води найчастішими легкими поливами є найменш бажаним способом поливу. Цей метод ніколи не допускає зволоження профілю ґрунту на достатню глибину, що сприяє дрібному зростанню коріння. Це призводить до того, що рослини легко ушкоджуються і залежить від частого поливу. Норми поливу повинні змінюватись в залежності від типу рослин, ґрунту та клімату.

## 1.4 Мета та завдання роботи

Метою даної роботи є розробка системи автоматичного управління (САУ) зрошенням ґрунту.

Проведений аналіз показав:

1) Існує багато різновидів систем зрошення, вибір яких визначається залежно від зрошуваної площі, особливостей рельєфу, наявності водних та енергетичних ресурсів;
2) Планування періодичності та часу зрошення обумовлено складом ґрунту, кліматичними умовами та безліччю факторів, які важко піддаються точному розрахунку;
3) Основним параметром, що визначає необхідність зрошення, є кількість доступної для рослин вологи;

4) Основним параметром, що характеризує роботу системи зрошення, є ефективність системи зрошення.

З урахуванням вище викладеного основними завданнями роботи є:

1) Розробка пристрою вимірювання та передачі параметрів (ПВП), що забезпечує дистанційне вимірювання кількості вологи, доступної для рослин та параметрів навколишнього середовища, що впливають на швидкість її зміни.
2) Розробка пристрою управління періодичністю та часом зрошення (ПКЗ) на основі показань ПВП.
3) Розробка програмного забезпечення для моніторингу параметрів, та управління налаштуваннями системи.
4) Виготовлення дослідних зразків та апробація системи автоматичного керування зрошенням ґрунту в польових умовах.

Враховуючи, що розміри сільськогосподарських полів можуть бути довжиною понад 1 км, а також їх віддаленість від електропостачання, пристрої системи повинні мати відповідний радіоканал передачі даних, а також автономне електропостачання на сонячних батареях.

# 2 РОЗРОБКА ПРИСТРОЮ ДЛЯ ВИМІРЮВАННЯ ТА ПЕРЕДАЧІ ПАРАМЕТРІВ

## 2.1 Методи вимірювання вологості ґрунту

Гравіметричні методи вимірювання вологості ґрунту базуються на вилученні води із зразка шляхом випаровування, вимивання або хімічної реакції. Кількість видобутої із зразка води вимірюється, і на цій основі розраховується вологість ґрунту. Вимірювання кількості води відбувається декількома методами. Найпростіший метод – вимірювання зменшення ваги зразка. Вимірювання кількості видобутої води також може проводитися шляхом дистиляції або всмоктування осушувачем. Нарешті, вміст води у зразку може бути визначено кількісним виміром продуктів реакції, вилучених із зразка. При кожному з цих методів відбувається поділ ґрунту та води з вимірюванням або оцінкою обсягу видобутої води.

Для вимірювання вологості ґрунту за допомогою сушильної шафи потрібні контейнери із щільно прилеглими кришками, шафи, де можна контролювати температуру та ваги. Сушильна шафа може бути конвективною або з примусовою вентиляцією. Найточніші дані дає вакуумна сушильна шафа. Вимірювання вологості ґрунту в сушильній шафі вважається стандартом точності. А ось визначення вологості поверхневого шару ґрунту утруднене через знижену чутливість. Набір обладнання дуже простий: пробовідбірник, ваги та сушильна шафа. У той же час вимір вологості цим методом вимагає багато часу: зразок висушується приблизно добу. При цьому вологість ґрунту легко обчислити за масою. Визначення вологості ґрунту гетерогенного профілю утруднено, як і визначення вологості на певній глибині.

Тензіометричний метод – найвідоміший метод виміру вологості ґрунту базується на здатності останньої вбирати воду. Нуль на шкалі тензіометра означає, що грунт повністю насичений вологою. У той же час максимальний

показник тензіометра – 1 бар. Таким чином, діапазон вологості ґрунту, в якому тензіометр може працювати, обмежений. І в ґрунтах високої потенційної вологоємності або в дуже сухих умовах тензіометри зашкалювали та ламалися.

Тензіометр вимірює водопоглинаючу здатність ґрунту, а вологість ґрунту – побічно. Прямі вимірювання на поверхні ґрунту цим методом неможливі. Інформацію про насиченість ґрунту вологою можна отримати майже в режимі реального часу. Реакція системи на зміни характеристик ґрунту дуже швидка. Прилади легко розміщуються у ґрунті. При калібруванні приладів необхідно розуміти водопоглинаючу здатність різних типів ґрунтів. Вартість приладів порівняно низька.

Електричний метод заснований на вплив води на електричні властивості ґрунту. Вологість ґрунту впливає на електричний опір ґрунту. Однак неоднорідність ґрунту заважає вимірюванню опору прямими методами. Багато проблем вимірювання електричного опору ґрунту вирішують пористі блоки. Ці однорідні блоки, що містять вбудовані електроди, поміщають у ґрунт до досягнення рівноважної із ґрунтом вологості. Тоді їх електричні характеристики беруть за однозначні з характеристиками ґрунту. Однак опір таких пористих блоків залежить від концентрації іонів у воді, тому цей метод не дозволяє досягти високої точності вимірювання, а іноді похибка може становити 100%. Метод не дозволяє виміряти вміст зв'язаної води, особливо на глинистих ґрунтах. Крім того,

Методи вимірювання діелектричної проникності ґрунту. Успішна поляризація молекул води без поляризації розчинених у воді іонів залежить від того, як швидко відбувається ця поляризація, або від частоти вимірювання. На нижчих частотах діелектричні датчики поляризують воду та солі, що робить їх чутливими до солоності ґрунту. Однак із збільшенням частоти виміру (більше 50 МГц) цей вплив зменшується. Таким чином, якщо датчик працює в діапазоні кГц, датчик не може уникнути багатьох факторів,

що знижують точність вимірювань. І, навіть якщо датчик працює на високій частоті виміру, це ще не гарантує успіху. Важливу роль також відіграє правильне проектування електричної системи.

Є кілька типів діелектричних датчиків. Найбільш поширені на ринку датчики вмісту води поділяються на дві основні категорії. Ємнісний датчик – використовує ґрунт як елемент конденсатора і використовує ємність накопичення заряду ґрунту для калібрування за вмістом води. Рефлектометрія у часовій області – вимірює час проходження відбитої хвилі електричної енергії вздовж лінії передачі. Час у дорозі залежить від ємності ґрунту та об'ємного вмісту води. Цікаво, що подібні датчики використовують кілька частот у сигналі, що допомагає зменшити помилки, пов'язані із засоленням ґрунту. Кореляція з гравіметричним методом виміру вологості перевищує 0,9. Досягається точність виміру 2%. Обладнання дуже складне, проте сам аналіз дуже простий і може бути проведений менш ніж за 5 секунд. Прилад може утримувати у пам'яті результати аналізів тривалий час. Метод дозволяє досягти дуже високої точності, але дуже дорогий.

Мікрохвильові методи засновані на вимірі інтенсивності відбитих від поверхні ґрунту електромагнітних хвиль. Показники відбиття грунтом залежать від його вологості. Відображення електромагнітних хвиль поверхнею ґрунту в мікрохвильовому діапазоні можна визначити дистанційно відповідними вимірювальними приладами – пасивними (радіометричними) або активними (радар) методами. Дозвіл пасивних систем обмежений розмірами антени. Робота активних систем базується на тому, що здатність ґрунту розсіювати мікрохвильове випромінювання залежить від його вологості, нерівності поверхні та електропровідності. Рослинний покрив знижує потужність відбитого випромінювання до 40%. Обмежуючим фактором є здатність методу вимірювати вологість ґрунту лише у верхньому шарі завглибшки кілька сантиметрів. Таким чином,

Термічний метод базуються на зв'язку теплової інерції ґрунту та його вологості. Використання методу утруднене поглинанням сонячної енергії

внаслідок випаровування води з поверхні ґрунту. Випаровування також знижує добову амплітуду коливань температури поверхні ґрунту. Таким чином, різниця денної та нічної температури відображає вологість ґрунту та випаровування з його поверхні. Численні дослідження показали, що для певних ґрунтів добові коливання температури поверхні є хорошим індикатором вмісту вологи у верхньому (до глибини 4 см) шарі ґрунту. У той же час цей метод не підходить для полів, покритих рослинним покривом. Також він залежить від ґрунтово-кліматичних умов і здебільшого працює лише у поверхневому шарі ґрунту.

Метод поляризованого світла базується на тому, що за наявності вологи на поверхні світло, відбите від неї, поляризується. Близький інфрачервоний метод заснований на поглинанні молекулами води поверхневому шарі інфрачервоного випромінювання на певних частотах. Технологія забезпечує швидке проведення вимірювань, але залежить від нерівності поверхні та показує лише поверхневу вологість.

Метод нейтронного розсіювання. Середнє значення втрат енергії значно вище, коли нейтрони стикаються з атомами з нижчою атомною вагою. У ґрунті атоми з малою вагою представлені переважно воднем. В результаті водень уповільнює швидкі нейтрони набагато ефективніше, ніж будь-який інший елемент у ґрунті. Оскільки найбільшим джерелом атомів водню у ґрунті є вода, існує зв'язок між вологістю ґрунту та втратами енергії. Прилади, які використовують цей метод, використовуються на практиці дуже рідко через високу вартість обладнання.

## 2.2 Розробка датчика для вимірювання вологості ґрунту

Як було показано в розділі 1.2, близько 50% ґрунту складається з твердих мінералів та органічних речовин, інші 50% займає повітря і вода. Також відомо, що теплопровідність повітря становить 0.026Вт/(м*К), а теплопровідність води становить 0.6Вт/(м*К).Для аналізованого ґрунту

теплопровідність сухого ґрунту буде близько 0.2Вт/(м*К), а ґрунту, насиченому вологою близько0.5Вт/(м*К). Різниця в теплопровідності сухого та вологого ґрунту більш ніж удвічі може бути використана для визначення вологості ґрунту.

В основу конструкції датчика покладено метод визначення вологості ґрунту за часом дисипації теплової енергії. Розроблений метод не має аналогів, а конструкція датчика опублікована у відкритих джерелах інформації (1) та (2).

Як нагрівач обраний малопотужний кремнієвий транзистор 2N2222 (технічні характеристики транзистора наведено в Додатку А).

Для вимірювання температури використовувався цифровий термометр DS18B20 (технічні характеристики термометра наведено у Додатку А).

Поверхні нагрівача та термометра розташовані на відстані близько 1 мм та змонтовані на платі розміром 20х7 мм.

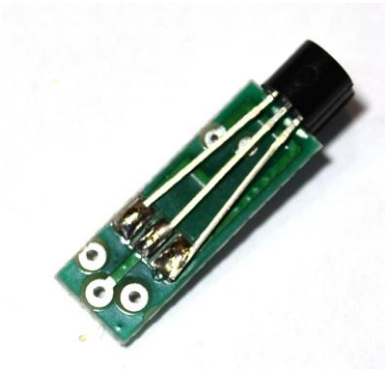


Рисунок 2.6 – Зовнішній вигляд плати датчика DS18B20

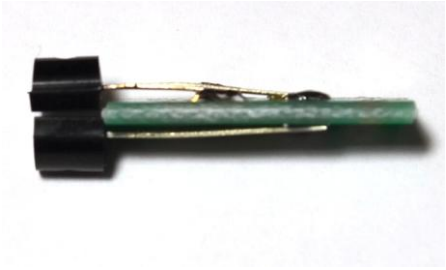


Рисунок  2.7 – Зовнішній вигляд плати датчика збоку

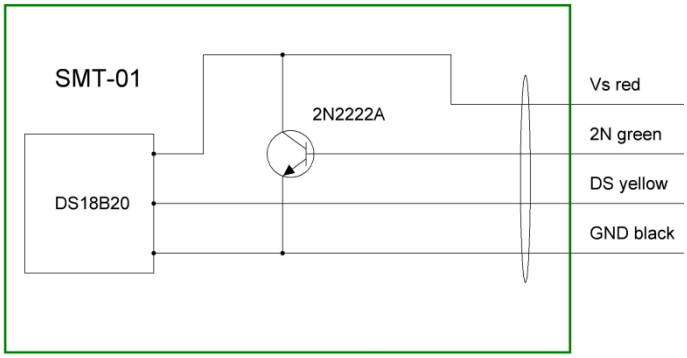



Рисунок 2.8 – Схема електричних з'єднань датчика

Плата з нагрівачем та термометром заливаються епоксидним компаундом для електричної ізоляції з'єднань та рівномірного відведення тепла у ґрунт. Об'єм корпусу датчика становить 2.5 мл. Для розробленого датчика було прийнято позначення SMT-01 (Soil Moisture and Temperature sensor).

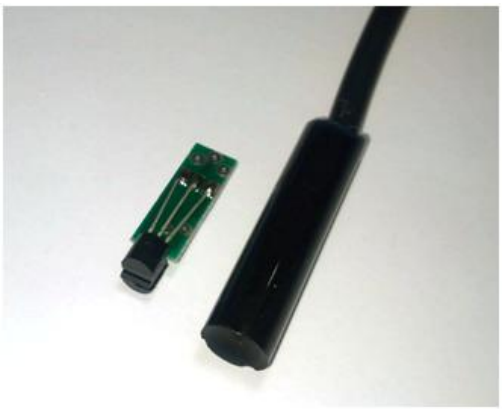


Рисунок 2.9 – Датчик SMT-01 після заливання епоксидним компаундом

Електрична схема підключення датчика SMT-01 до мікроконтролера (МК) Arduino наведена на рисунку 2.10.

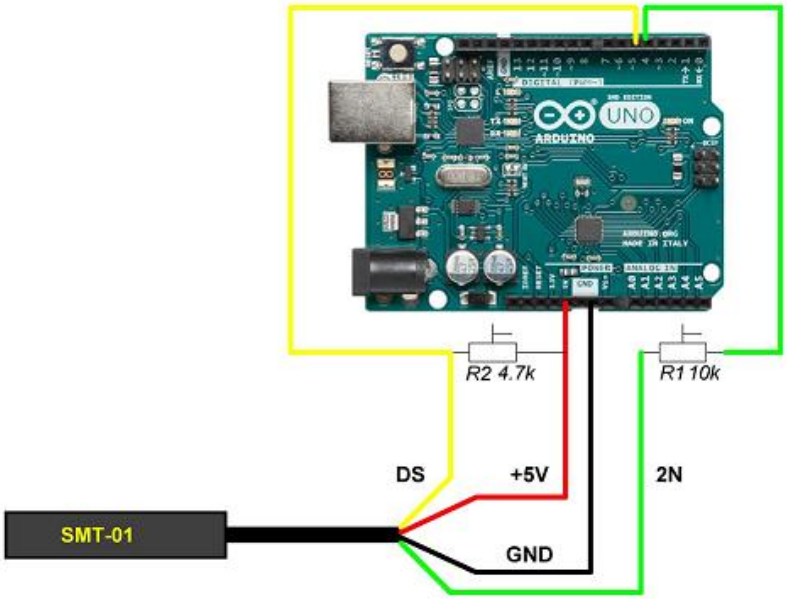



Рисунок 2.10 – Електрична схема підключення датчика SMT-01

Резистор R1 служить регулювання струму через нагрівач. А резистор R2 виконує функцію «підтягуючого резистора» в ланцюзі передачі даних протоколу 1-Wire.

Алгоритм вимірювання включає чотири етапи:

1. Вимірювання початкової температури ґрунту Tsm;
2. Нагрів датчика фіксований час 60 секунд (температура датчика підвищується на 5 – 10 оС залежно від складу ґрунту);
3. Вимірювання часу дисипації теплої енергії до температури (Tsm – 1 оС).
4. Розрахунок вологості ґрунту.

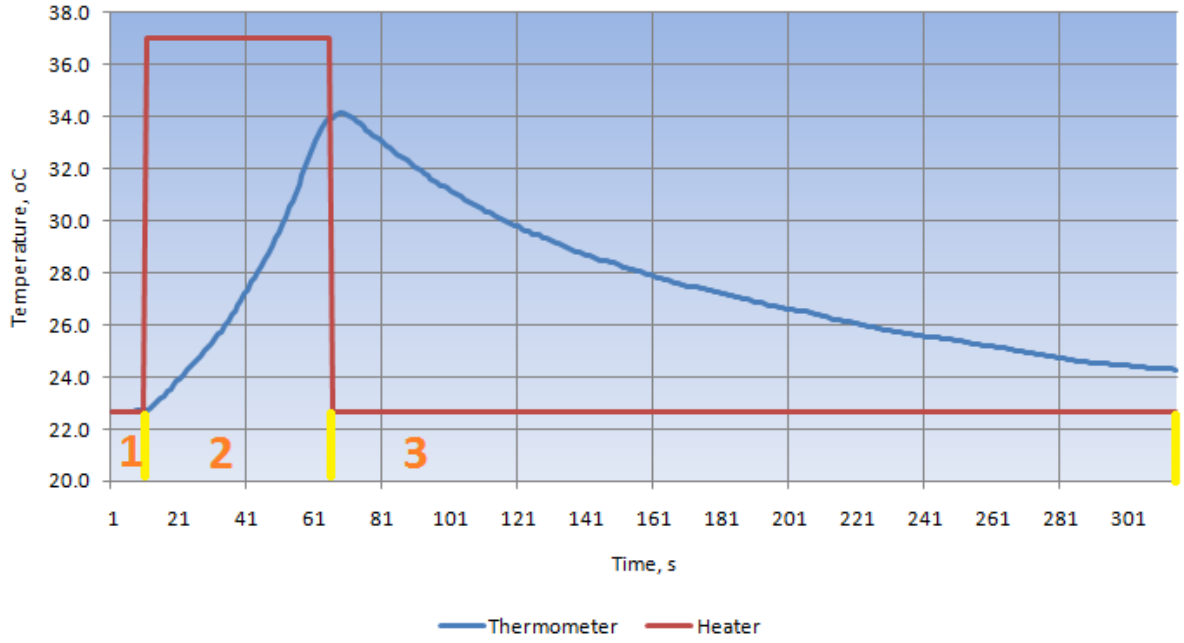



Рисунок 2.11 – Тимчасова діаграма роботи датчика SMT-01

Програмне забезпечення для МК, що реалізує алгоритм вимірювання температури та вологості ґрунту, наведено в Додатку Б (Програма для вимірювання температури та вологості ґрунту).

Датчик SMT-01 пройшов випробування у різних за складом ґрунтах. Для випробувань використані: універсальна суміш для рослин на основі торфу, суглинок та річковий пісок. Ґрунт до початку випробувань ретельно висушувався. Об'єм ґрунту при проведенні випробувань – 500 мл. Вода додавалася порціями по 10 мл кожні 30 хвилин до повного насичення ґрунту водою. Результати випробувань подано на рис. 2.12.

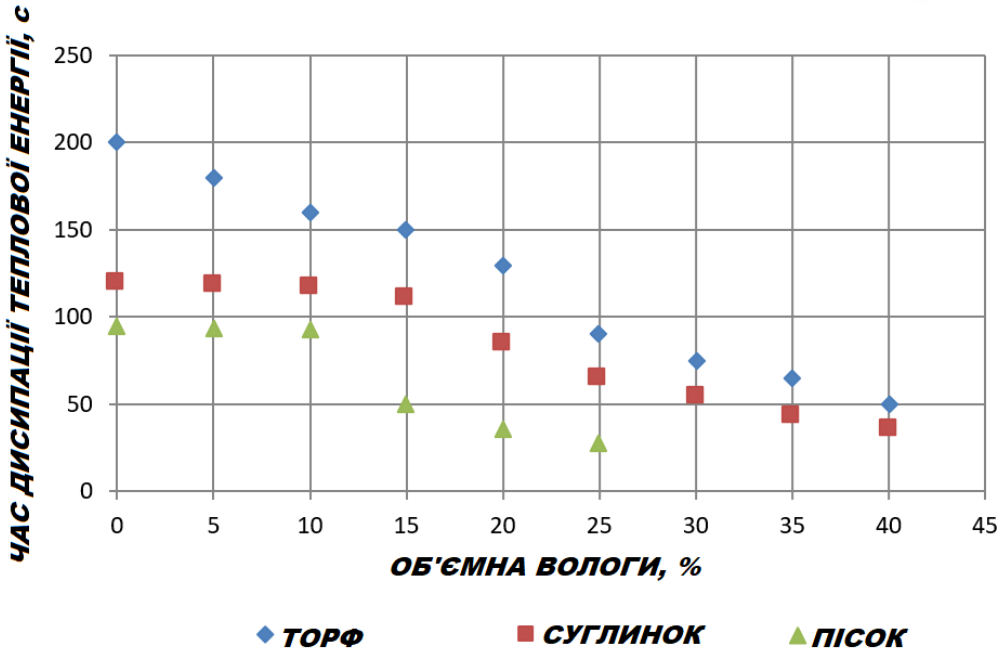



Рисунок 2.12 – Залежність часу дисипації теплової енергії датчика SMT-01 від об'ємної вологості до різних складів ґрунту

Результати випробувань датчика SMT-01 дозволяють зробити такі основні висновки:

1) Датчик SMT-01 можна використовувати для вимірювання вологості різних за складом видів ґрунту;
2) Як основний параметр для управління системою зрошення слід використовувати параметр AW, так саме цей параметр найбільш точно відображає кількість вологи, яку може використовувати рослина.

## 2.3 Датчики для вимірювання параметрів довкілля

Крім вологості та температури ґрунту, такі параметри навколишнього середовища як:

1) Температура повітря;
2) вологість повітря;

3) освітленість;

- є важливими для роботи системи автоматичного поливу рослин.

Для вимірювання температури та вологості повітря вибрано датчик HTU20D. Детальні технічні характеристики датчика наведено у Додатку А. Датчик має I2C інтерфейс та живлення 3.0-3.6V, що робить його зручним при використанні з МК Arduino.

Для монтажу датчика було розроблено пластиковий корпус та 3D модель.

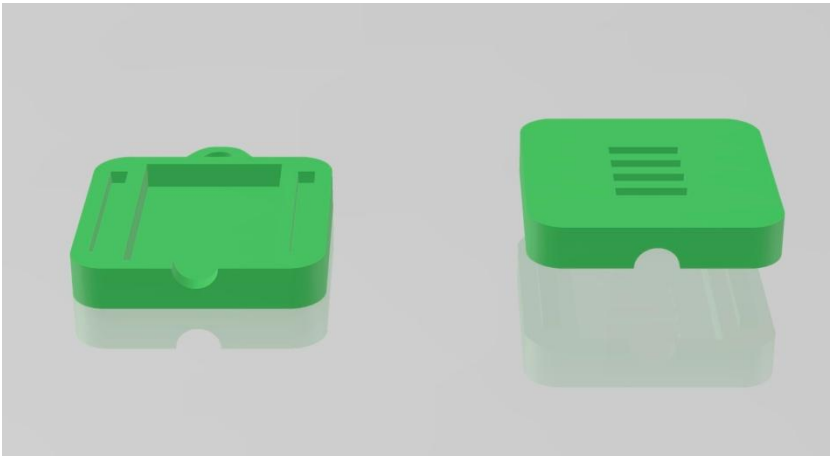


Рисунок 2.13 – 3D-модель корпусу датчика температури та вологості повітря

Зразок корпусу датчика було роздруковано на 3D принтері (рис. 2.14).

Рисунок 2.14 Датчик температури та вологості повітря.

Для вимірювання освітлення вибрано фоторезистор GL55. Детальні технічні характеристики датчика наведено у Додатку А. Датчик живиться від напруги 5V і підключається до МК через резистивний дільник до входу аналого-цифрового перетворювача (АЦП). Датчик GL55 вимагає тарування.

Для монтажу датчика було розроблено пластиковий корпус та його 3D модель (рис. 2.15). Зразок корпусу датчика було роздруковано на 3D принтері.

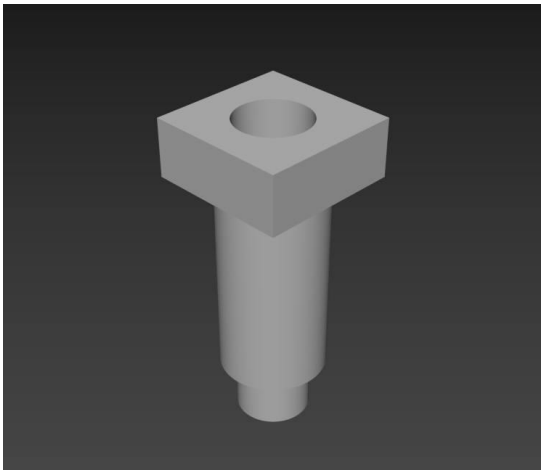


Рисунок 2.15 – Корпус датчика освітленості

## 2.4 Забезпечення передачі

При виборі способу передачі між пристроями системи враховані такі основні фактори:

1) дальність передачі;
2) низьке енергоспоживання;
3) невисока вартість компонентів;
4) частота та потужність передавача.

З огляду на це найбільш підходящим для забезпечення передачі даних між пристроями системи є протокол LoRa® (Long Range) – це технологія

модуляції сигналу фізичного рівня, орієнтована на пристрої живлення від батареї та передачу сигналу на великі відстані.

Технологія LoRa розроблена компанією Semtech і використовує запатентований метод модуляції сигналу з розширенням спектра, де дані кодуються широкосмуговими імпульсами з частотою, що змінюється. Застосування такого методу має такі переваги: значно підвищується чутливість приймача (до -148дБм) та знижується критичність до розбіжності за частотою між приймачем та передавачем. Технологія LoRa забезпечує зв'язок на великі відстані (до 20 км прямої видимості), низьке енергоспоживання, високу чутливість приймача, низькі швидкості передачі даних та захищену передачу із шифруванням.

Як приймач-передавач обраний модуль Ai-Thinker Ra-02 LoRa з робочою частотою 433 MHz, виконаний на основі чіпа компанії Semtech SX1278. Частота і потужність приймача не вимагають спеціальної ліцензії.

## 2.5 Опис електричної схеми

Принципова електрична схема пристрою вимірювання та передачі даних (ПВП) показана на рис. 2.16.

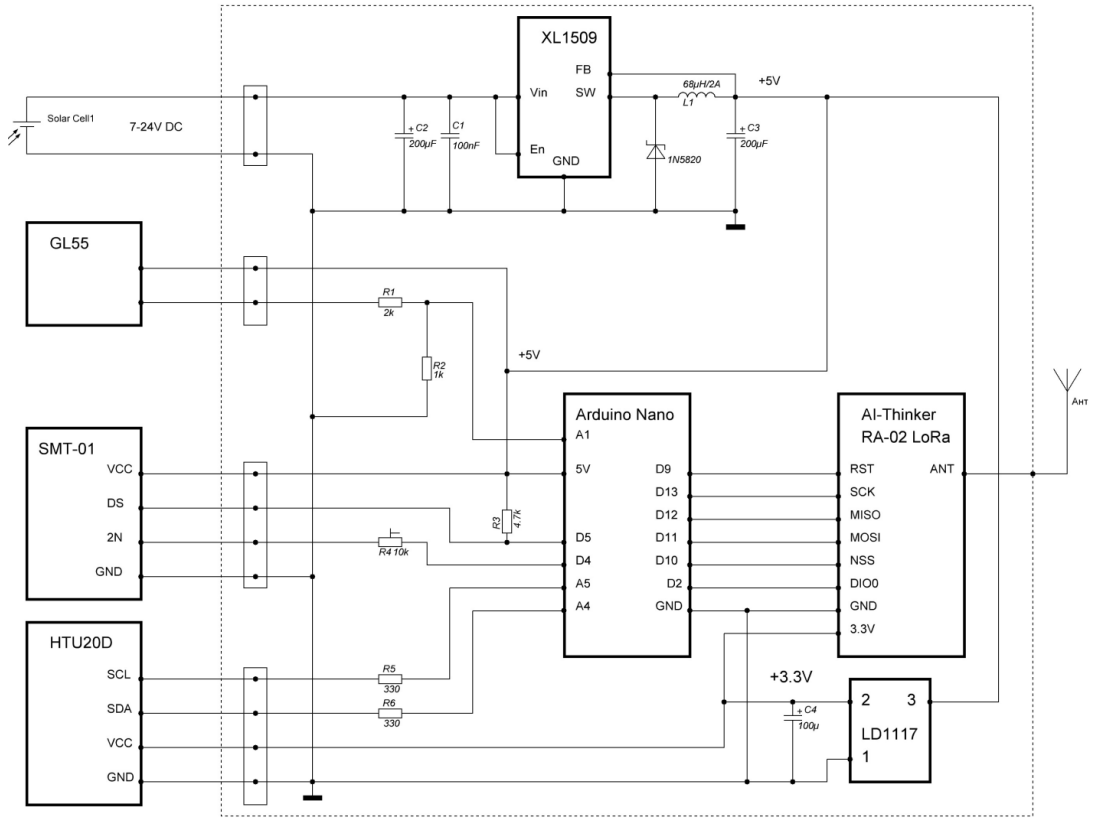



Рисунок 2.16 – Принципова електрична схема пристрою вимірювання та передачі параметрів

Електрична схема пристрою включає такі основні вузли:

1) Мікроконтролер Arduino Nano;
2) Приймач-передавач AI-Thinker R-02 LoRa 433 MHz;
3) Перетворювач постійної напруги XL1509;
4) Датчик вологості та температури ґрунту SMT-01;
5) Датчик температури та вологості повітря HTU20D;
6) Датчик освітлення GL55.

Дільник напруги на резисторах R1 та R2 забезпечує підключення датчика освітленості GL55 до аналогового входу A1 Arduino Nano.

Резистор R3 грає роль підтягуючого резистора в ланцюзі передачі сигналу протоколу 1-Wire відповідно до рекомендацій виробника цифрового датчика температури DS18B20.

Змінний резистор R4 призначений для регулювання струму нагрівача датчика SMT-01.

Резистори R5 та R6 призначені для узгодження рівнів логічного сигналу датчика HTU20D та Arduino Nano.

Перетворювач напруги, виконаний на чіпі XL1509, забезпечує пристрій стабілізованою напругою 5В та струмом до 2А.

Живлення приймача-передавача та датчика HTU20D здійснюється напругою 3.3В від стабілізатора напруги LD1117.

Список компонентів електричної схеми пристрою наведено у Таблиці 2.1.

Таблиця 2.1 Пристрій вимірювання та передачі параметрів

| Найменування компонента | Кільк. |
|---|---|
| Arduino Nano | 1 |
| AI Thinker RA-2 LoRa | 1 |
| Адаптер подовжувач IPX (U.FL) SMA – 15 см. | 1 |
| Антена 5dbi/14.5CM SMA 433 МГц | 1 |
| Індуктивність SUMIDA CDRH127NP-330MC | 1 |
| Конденсатор CAP-220mkf – 50v (105°C) SMD(RC) 10*10 | 2 |
| DC/DC перетворювач XL1509 | 1 |
| Конденсатор чіп кераміка (0805) 0,1mkf (X7R) 50v 10% | 1 |
| Діод Шоттки 1N5820 | 1 |
| SMD резистор (0805) 2 kom ±1% | 1 |
| SMD резистор (0805) 1 kom ±1% | 1 |
| SMD резистор (0805) 4.7 kom ±5% | 1 |
| SMD резистор (0805) 330 om ±5% | 2 |
| Потенціометр багатооборотний металокерамічний 3266Y-10K | 1 |
| Клемник DEGSON DG500-5.08-02P-14-00AH | 6 |

Продовження таблиці 2.1

| Найменування компонента | Кільк. |
|---|---|
| DC/DC перетворювач LD1117AG-33-AA3 SOT223 | 1 |
| Конденсатор ECAP-GS-100mkf – 25v (GS) | 1 |
| Друкована плата | 1 |
| Корпус G212MF-IP67 | 1 |
| Кабельне введення Rittal 2411.806 | 4 |
| Датчик температури та вологості SMT-01 | 1 |
| Датчик температури та вологості повітря HTU21D-MOD | 1 |
| Датчик освітленості GL55 | 1 |

## 2.6 Розробка прототипу пристрою

Для апробації та налагодження електричної схеми було зібрано прототип пристрою (рис. 2.17).

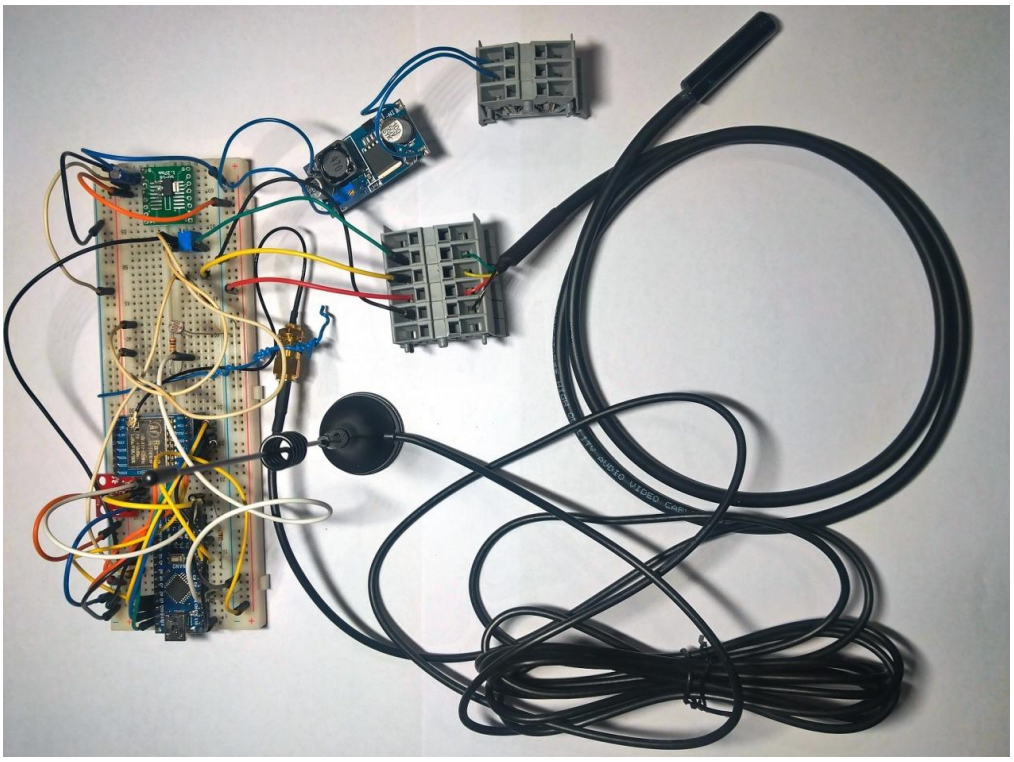


Рисунок 2.17 – Прототип пристрою

Струм, споживаний пристроєм, трохи більше 150 мА.

Для монтажу компонентів електричної схеми пристрою було розроблено друковану плату. Рознімання на друкованій платі призначені для підключення датчиків та джерела живлення пристрою.

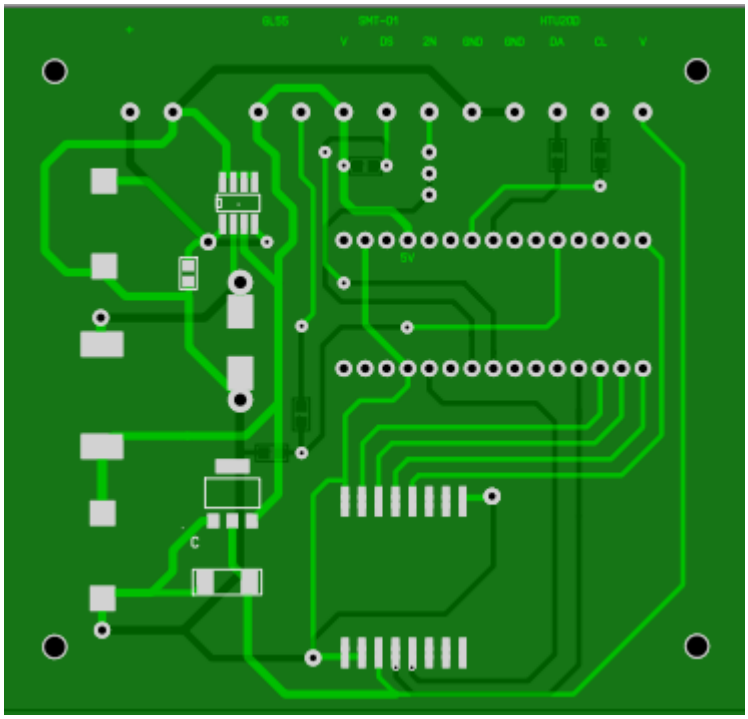


Рисунок 2.18 Друкована плата пристрою.

# 3 РОЗРОБКА ПРИСТРОЮ УПРАВЛІННЯ ЗРОШЕННЯМ

## 3.1 Функції та режими роботи керуючого пристрою

Пристрій керування зрошенням (ПКЗ) призначений для увімкнення виконавчих елементів системи зрошення (клапанів, насосів, пускових пристроїв, індикаторів) та може працювати в автоматичному або ручному режимі.

Залежно від конфігурації системи зрошення, можливі такі варіанти використання ПКЗ:

1) Робота в автоматичному режимі керування. ПКЗ отримує дані від ПВП. Якщо значення кількості доступної вологи AW нижче за допустимий рівень, то ПКЗ включає реле (електромагнітний клапан, насос або пусковий пристрій потужного насоса). Якщо значення AW досягає необхідного рівня, то УОО вимикає реле.
2) Робота в ручному режимі керування. На великій площі (сільськогосподарське поле) розташовані кілька ПВП, кожному їх відповідає своє ПКЗ. ПКЗ отримує дані від ПВП. Якщо значення AW нижче за допустимий рівень, то ПКЗ включає індикатор. Якщо значення AW досягає необхідного рівня, то УОО вимикає індикатор. Оператор стежить за станом індикаторів та керує системою зрошення (дощувальною установкою фронтального або кругового типу).

ПКЗ також може бути використане для керування будь-яким параметром, який вимірюється за допомогою ПВП. Наприклад, при використанні системи в тепличному господарстві пристрій може керувати температурою повітря (включати/вимикати нагрівач або систему кондиціювання), вологістю повітря та освітленням.

3.2 Опис електричної схеми

Принципова електрична схема УУ наведено на рис. 3.1.

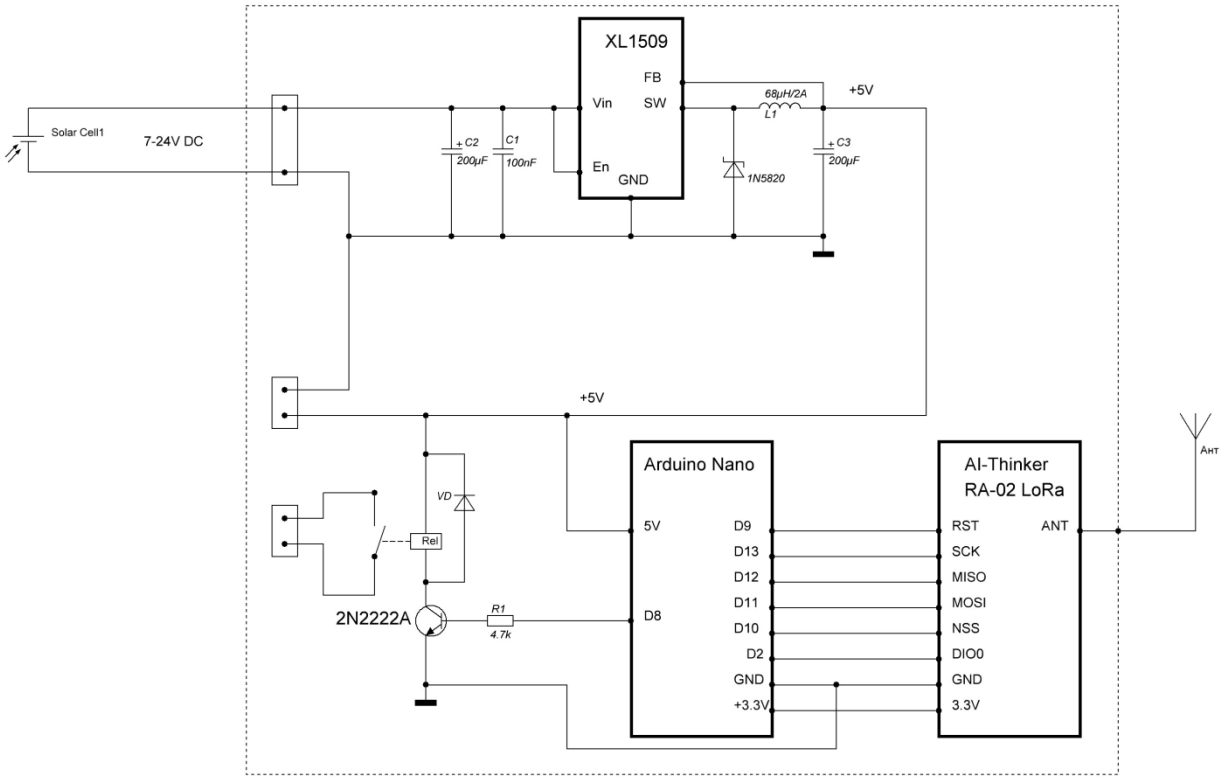



Рисунок 3.1 – Принципова електрична схема пристрою, що управляє

Електрична схема пристрою включає такі основні вузли:

1) Мікроконтролер Arduino Nano;
2) Приймач-передавач AI-Thinker R-02 LoRa 433 MHz;
3) Перетворювач постійної напруги XL1509;
4) Реле.

Для управління реле у схему включений транзистор 2N2222A. Діод VD захищає схему від зворотного струму індукції при вимиканні реле.

Перетворювач напруги, виконаний на чіпі XL1509, забезпечує пристрій стабілізованою напругою 5В та струмом до 2А.

Список компонентів електричної схеми пристрою наведено у Таблиці 3.1.

Таблиця 3.1 Керуючий пристрій

| Найменування компонента | Кільк. |
|---|---|
| Arduino Nano | 1 |
| AI Thinker RA-2 LoRa | 1 |
| Адаптер подовжувач IPX (U.FL) – SMA – 15 см. | 1 |
| Антена 433 МГц, 5dbi/14.5CM SMA | 1 |
| Індуктивність SUMIDA CDRH127NP-330MC | 1 |
| Конденсатор SMDCAP-220mkf – 50v (105°C) SMD(RC) 10*10 | 2 |
| DC/DC перетворювач XL1509 | 1 |
| Конденсатор Чіп кераміка (0805) 0,1mkf (X7R) 50v 10% | 1 |
| Діод Шотткі 1N5820 | 1 |
| SMD-резистор (0805) 4.7 kom ±5% | 1 |
| Транзистор біполярний 2N2222A TO-92 | 1 |
| Реле електромеханічне HF3FD/005-HSTF | 1 |
| Діод випрямляючий SMD 1N4148W SOD-123 | 1 |
| Клемник DEGSON DG500-5.08-02P-14-00AH | 1 |
| Друкована плата | 1 |
| Корпус G212MF-IP67 | 1 |
| Кабельне введення Rittal 2411.806 | 3 |

Для апробації та налагодження електричної схеми було зібрано прототип пристрою (рис. 3.2).

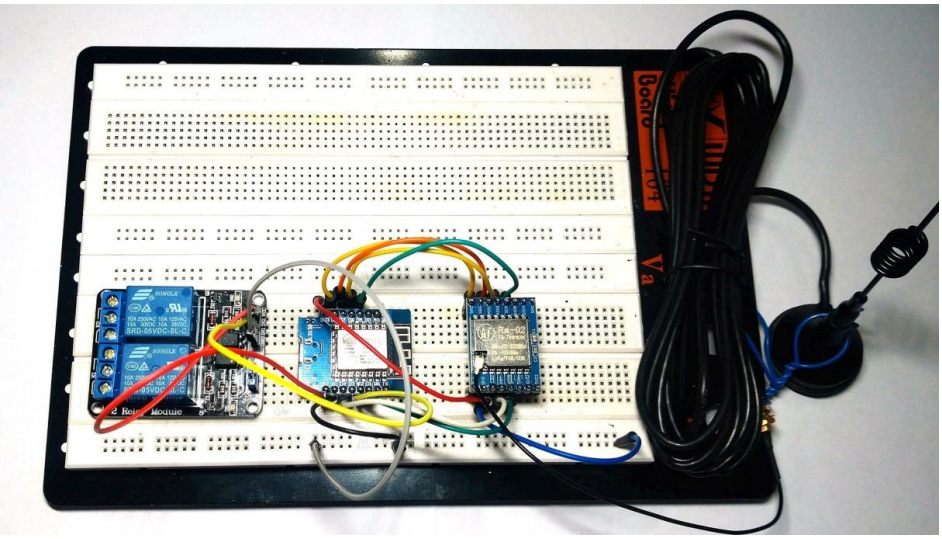


Рисунок 3.2 – Прототип пристрою

Струм, споживаний пристроєм, трохи більше 100 мА.

## 3.3 Прототип керуючого пристрою

Для монтажу компонентів електричної схеми пристрою було розроблено друковану плату. Рознімання на друкованій платі призначені для підключення датчиків та джерела живлення пристрою.

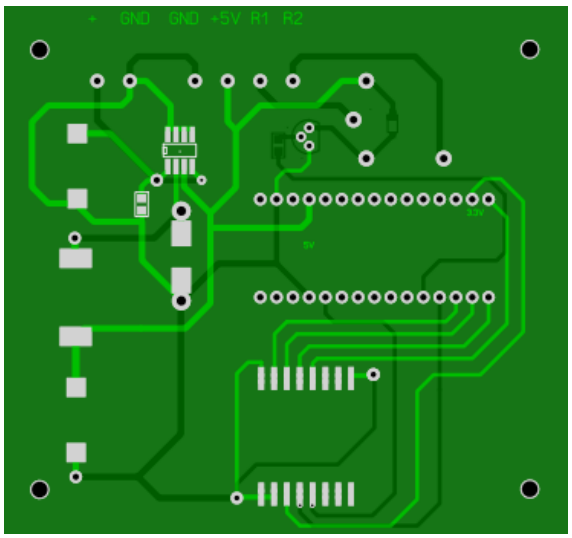



Рисунок 3.3 – Друкована плата пристрою

# 4 РОЗРОБКА ШЛЮЗУ ПЕРЕДАЧІ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 4.1 Алгоритм передачі

Обмін даними у системі проводиться пакетами у напівдуплексному режимі передачі. Кожен пакет даних передачі включає:

1) Адреса системи;
2) Адреса пристрою;
3) Адреса одержувача;
4) Довжина повідомлення;
5) Текст повідомлення (результати вимірювання, параметри налаштування, параметри команди тощо);
6) Рівень зв'язку.

Алгоритм прийому та обробки пакетів повідомлень наведено на рисунку 4.1. Алгоритм передачі пакетів повідомлень наведено на рисунку 4.2.

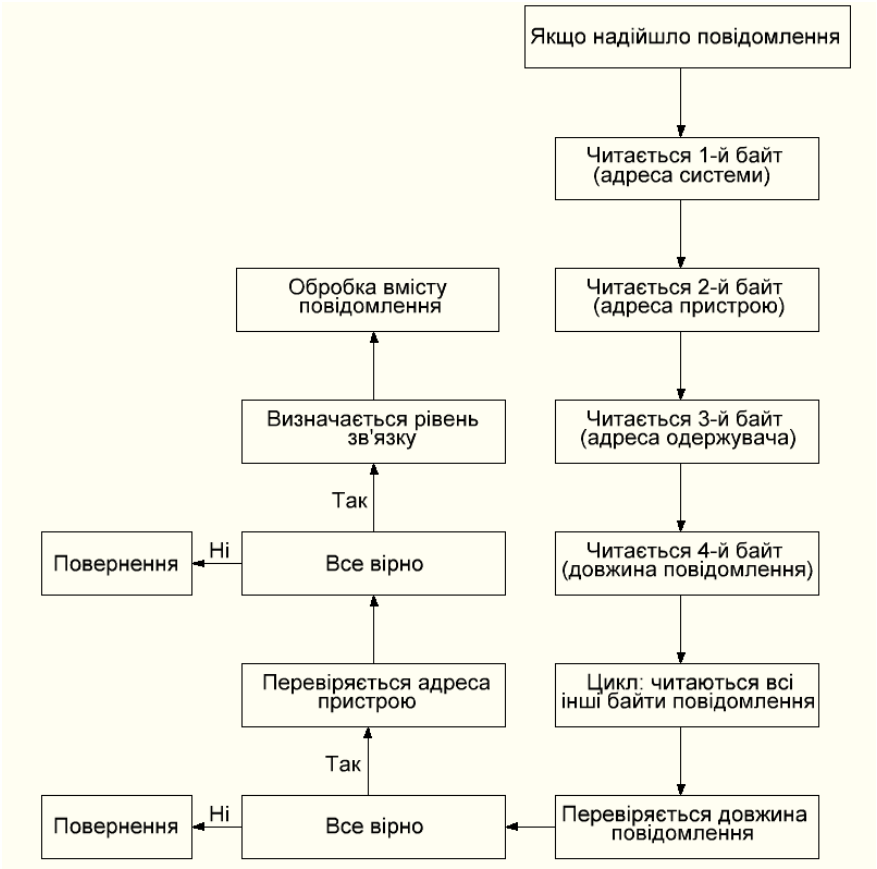



Рисунок 4.1 – Алгоритм прийому та обробки пакетів повідомлень

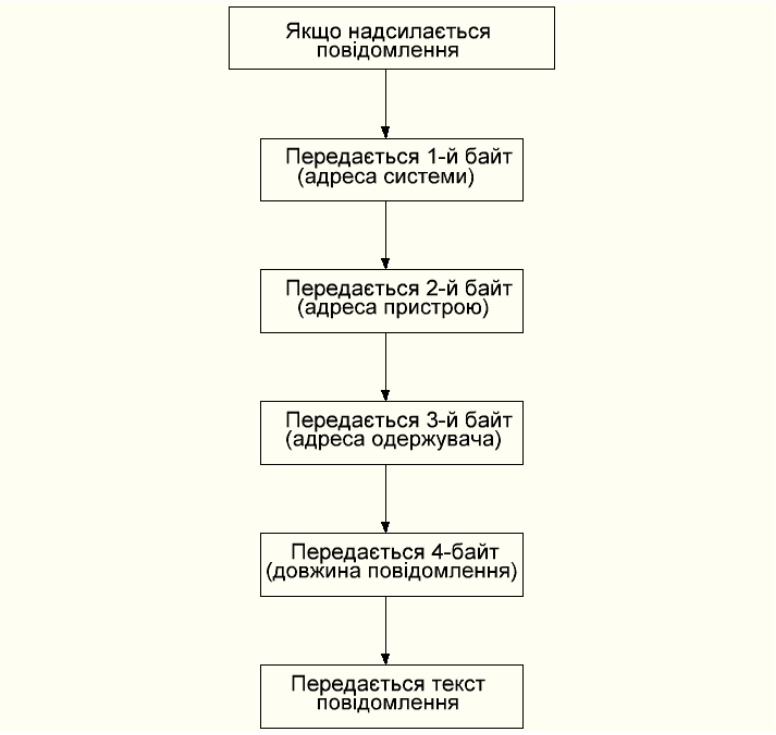



Рисунок 4.2 – Алгоритм передачі повідомлень

4.2 Розробка шлюзу передачі даних

Електричну схему шлюзу передачі (ШПД) показано на рис. 4.2.

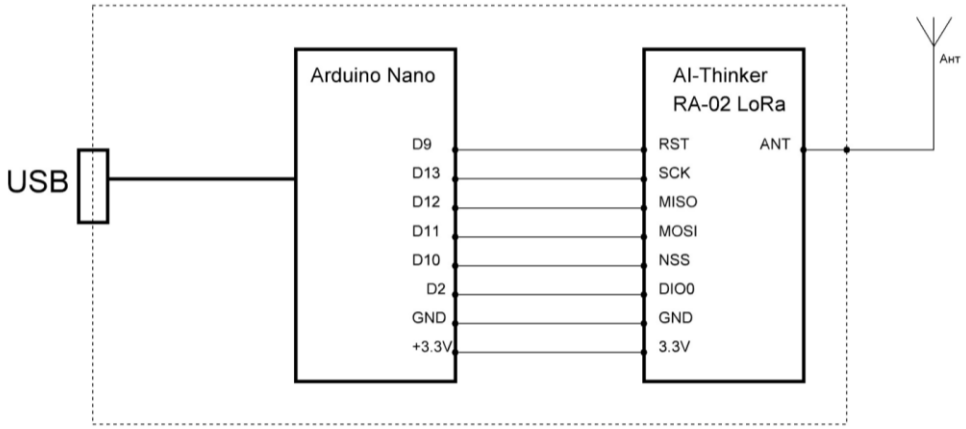



Рисунок 4.2 – Принципова електрична схема ШПД

Електрична схема ШПД включає такі основні вузли:

1) Мікроконтролер Arduino Nano;
2) Приймач-передавач AI-Thinker R-02 LoRa 433 MHz.

Живлення ШПД здійснюється через USB-порт від ПК.

Список компонентів електричної схеми ШПД наведено у Таблиці 4.1.

Таблиця 4.1 Шлюз передачі даних

| Найменування компонента | Кільк. |
|---|---|
| Arduino Nano | 1 |
| AI Thinker RA-2 LoRa | 1 |
| Адаптер подовжувач IPX (U.FL) SMA – 15 см. | 1 |
| Антена 5dbi/14.5CM SMA 433 МГц | 1 |
| Друкована плата | 1 |
| Корпус NUB1057020 | 1 |

Для апробації та налагодження електричної схеми було зібрано прототип пристрою (рис. 4.3).

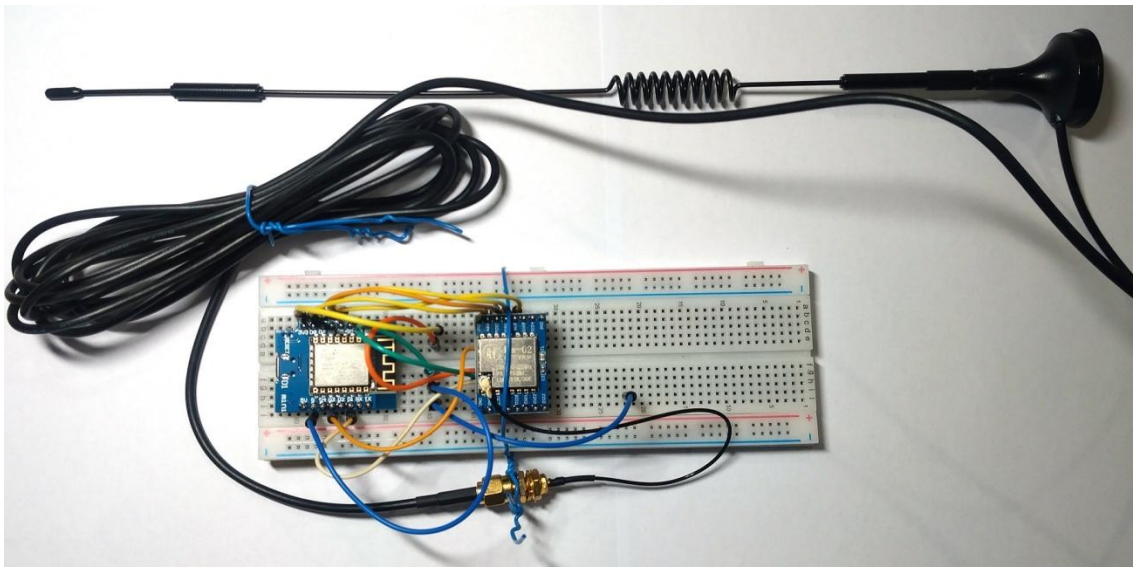


Рисунок 4.3 – Прототип ШПД

Струм, споживаний пристроєм, трохи більше 100 мА.

Для монтажу компонентів електричної схеми пристрою було розроблено друковану плату. Рознімання на друкованій платі призначені для підключення датчиків та джерела живлення пристрою.

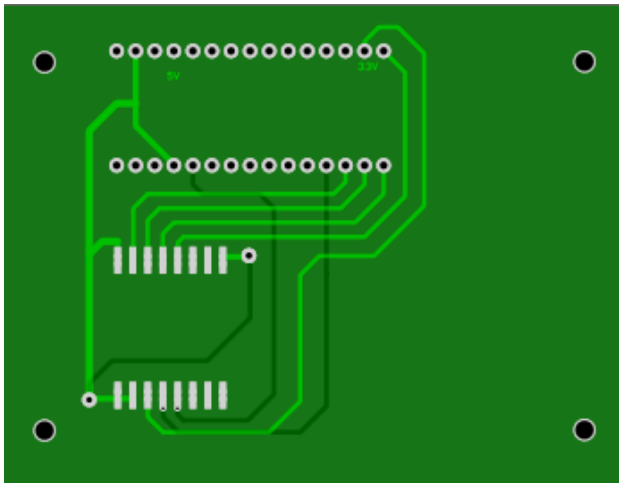


Рисунок 4.4 – Друкована плата шлюзу

## 4.3 Програмне забезпечення пристрою вимірювання та передачі параметрів

Програмне забезпечення ПВП розроблено мовою Сі серед програмування Ардуино.

Програма складається з таких основних блоків:

1) Ініціалізація змінних та налаштувань пристрою;
2) Вимірювання параметрів ґрунту та навколишнього середовища;
3) Прийом-передача даних протоколу LoRa.

Усі налаштування пристрою зберігаються в незалежній пам'яті МК. При включенні пристрою (подачі живлення) виконується зчитування даних налаштування та проводиться ініціалізація змінних та протоколів передачі даних. Цей блок програмного забезпечення виконує функція Setup().

Основний цикл ПЗ Loop() забезпечує періодичний вимір параметрів (1 раз на 7 хвилин). Бібліотечні функції OneWire та функції void DS18B20_init(void), void DS18B20_measure(void) та void Measure_SMT (void) забезпечують вимірювання вологості та температури ґрунту датчиком SMT-01. Вимірювання температури та вологості повітря забезпечують бібліотечні функції SparkFunHTU21D. Після закінчення вимірювання параметри передаються протоколом LoRa з допомогою функції void sendMessage(String outgoing). У той час, коли вимірювання параметрів не виконується, в основному циклі працює функція onReceive(LoRa.parsePacket()), що забезпечує прийом повідомлень за протоколом LoRa. Якщо в повідомленні є команда зміни налаштувань даного пристрою, то команда обробляється, а прийняті дані з налаштування зберігаються в незалежній пам'яті.

Повний текст програмного забезпечення наведено в Додатку Б.

## 4.4 Програмне забезпечення пристрою керування зрошенням

Програмне забезпечення ПКЗ розроблено мовою Сі серед програмування Ардуино.

Програма складається з таких основних блоків:

1) Ініціалізація змінних та налаштувань ПКЗ;
2) Прийом-передача даних протоколу LoRa;
3) Обробка даних та увімкнення/вимкнення виконавчого пристрою.

Усі налаштування ПКЗ зберігаються в енергонезалежній пам'яті МК. При включенні пристрою (подачі живлення) виконується зчитування даних налаштування та проводиться ініціалізація змінних та протоколів передачі даних. Цей блок програмного забезпечення виконує функція Setup().

Основний цикл програмного забезпечення Loop() забезпечує постійний моніторинг повідомлень за допомогою функції onReceive(LoRa.parsePacket()), що забезпечує прийом повідомлень за протоколом LoRa.

При надходженні результатів вимірювання параметрів від ПВП проводиться їх порівняння із заданими значеннями. У разі виходу значення КДВ за допустимі межі проводиться автоматичне вмикання/вимкнення виконавчого пристрою. Після закінчення операції параметри ПКЗ передаються протоколом LoRa за допомогою функції void sendMessage(String outgoing).

При надходженні команди від оператора через ШПД проводиться ручне включення/вимкнення виконавчого пристрою. Якщо в повідомленні є команда зміни налаштувань даного пристрою, то команда обробляється, а прийняті дані з налаштування зберігаються в незалежній пам'яті.

Повний текст програмного забезпечення наведено в Додатку Б.

## 4.5 Програмне забезпечення шлюзу передачі даних

Програмне забезпечення ШПД розроблено мовою Сі серед програмування Ардуино.

Програма складається з таких основних блоків:

1) Ініціалізація змінних та налаштувань ШПД;
2) Прийом-передача даних протоколу LoRa;
3) Прийом передачі даних через UART.

Всі налаштування ШПД зберігаються в незалежній пам'яті МК. При включенні пристрою (подачі живлення) виконується зчитування даних налаштування та проводиться ініціалізація змінних та протоколів передачі даних. Цей блок програмного забезпечення виконує функція Setup().

Основний цикл ПЗ Loop() забезпечує постійний моніторинг повідомлень за допомогою функції onReceive(LoRa.parsePacket()), що забезпечує прийом повідомлень за протоколом LoRa та моніторинг повідомлень через UART за допомогою функції Serial.read().

При надходженні результатів вимірювання параметрів від ПВП або даних від ПКЗ ШПД передає інформацію через UART для подальшої обробки ПЗ ПК.

При надходженні команди оператора, що управляє, через UART, ШПД відправляє її пристроям системи за протоколом LoRa за допомогою функції void sendMessage(String outgoing).

Повний текст програмного забезпечення наведено в Додатку Б.

## 4.6 Програмне забезпечення ПК

Програмне забезпечення для ПК (ПЗ) розроблено серед LabView. ПЗ включає такі основні файли:

1) Програму GS_Control;
2) Програму GS_Dashboard;

3) Базу даних GSIOT;

4) Файл конфігурації пристроїв GS_Devices.

Програма GS_Control забезпечує інтерфейс із ШПД та можливість налаштування параметрів пристроїв системи. Зовнішній вигляд інтерфейсу користувача показано на рис. 4.5.

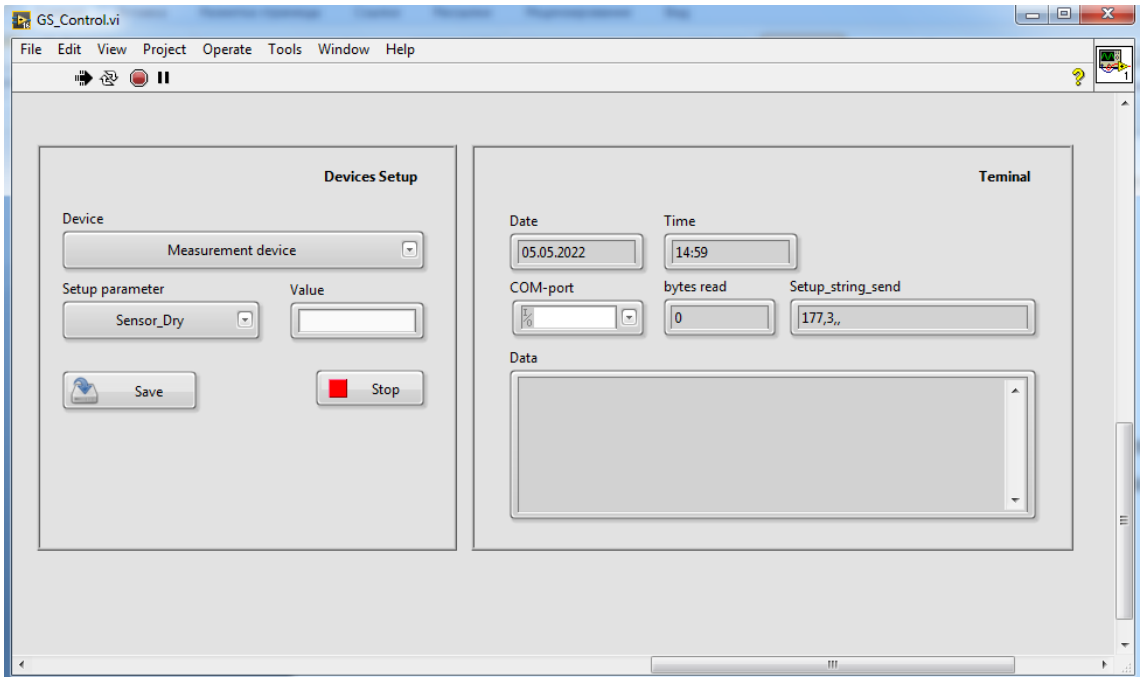



Рисунок 4.5 – Інтерфейс користувача програми GS_Control

Ліворуч розташована область Device Setup, призначена для виконання операцій з настроювання параметрів пристроїв системи. Селектор Device забезпечує вибір пристрою. Селектор Setup parameter призначений для вибору параметра. У полі Value вводиться нове значення параметра. За допомогою кнопки Save нове значення параметра записується у файл конфігурації та передається пристрою через ШПД.

Процес обміну даними з пристроями відображається в області Terminal.

Під час запуску програми виконується ініціалізація змінних (рис. 4.6).

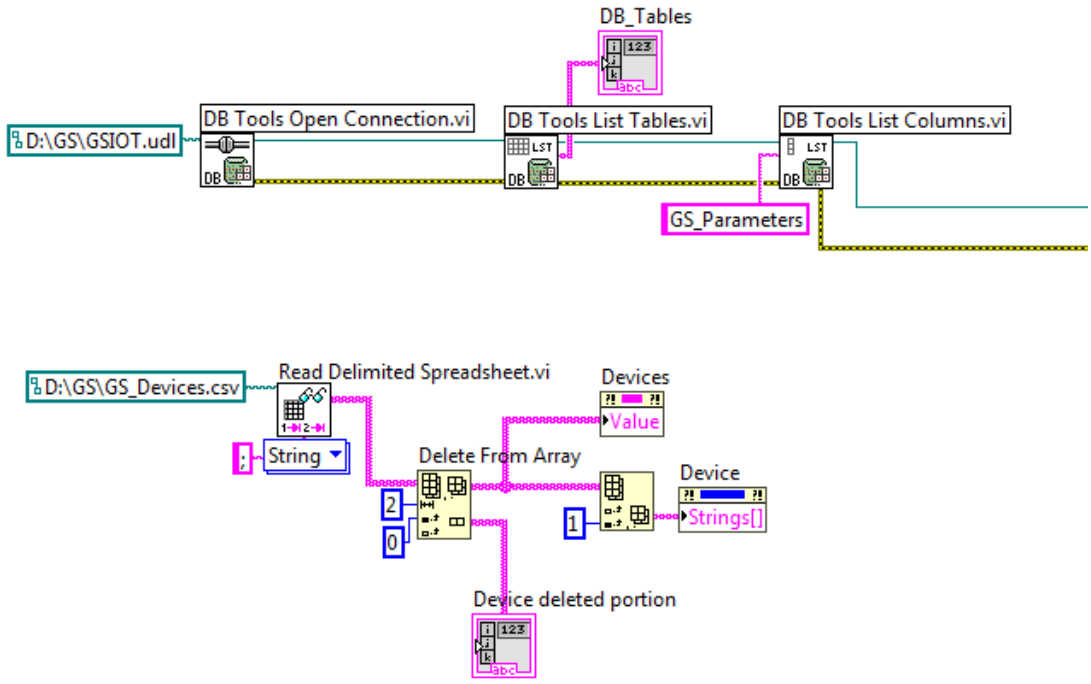



Рисунок 4.6 – Ініціалізація змінних під час запуску програми

Основний цикл програми складається з наступних основних блоків:

1) Читання повідомлень через COM порт (рис. 4.7);
2) Обробка повідомлень та запис до бази даних (рис. 4.8);
3) Передача повідомлень через порт COM (рис. 4.9).

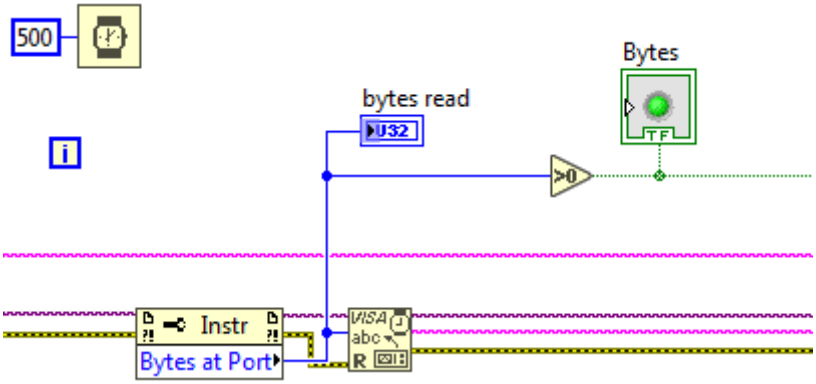



Рисунок 4.7 – Читання повідомлень через порт COM

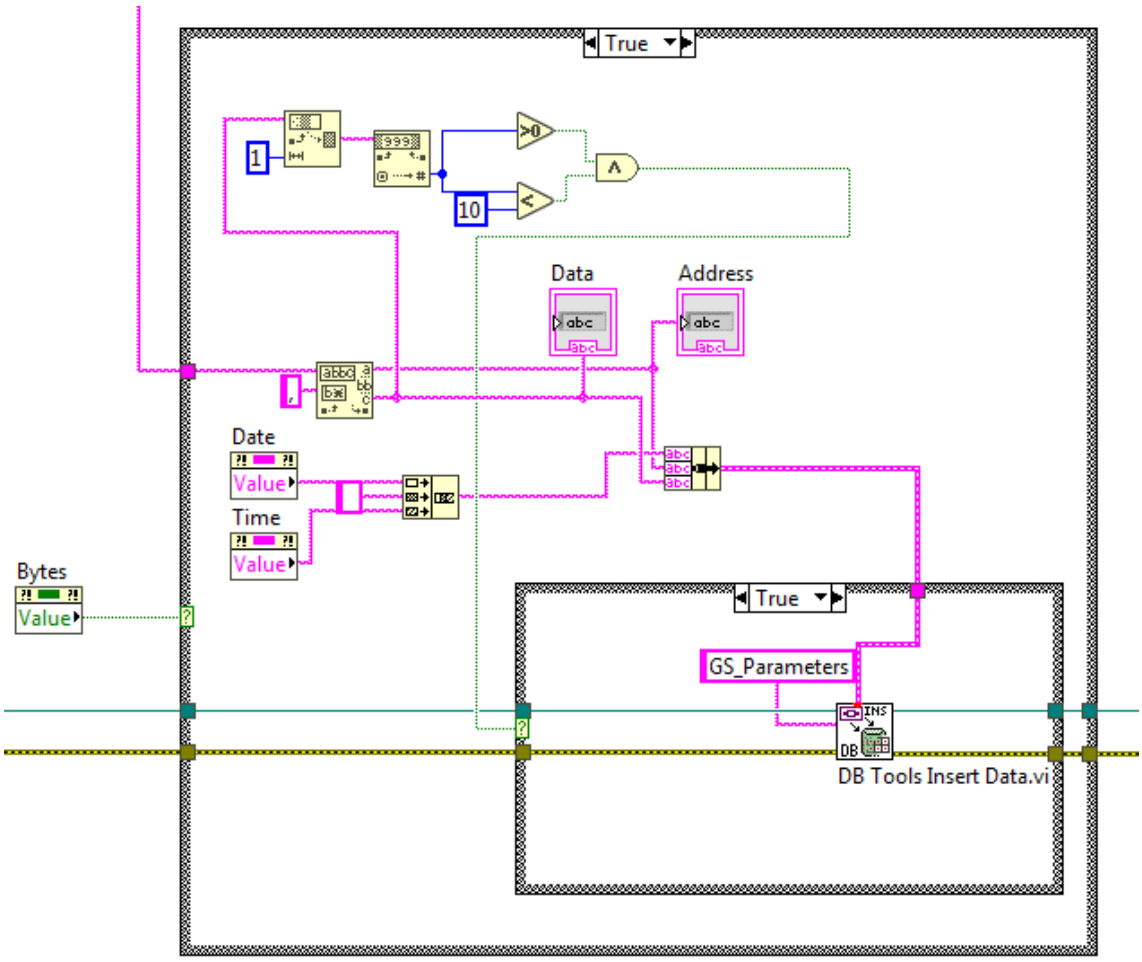



Рисунок 4.8 – Обробка повідомлень та запис до бази даних

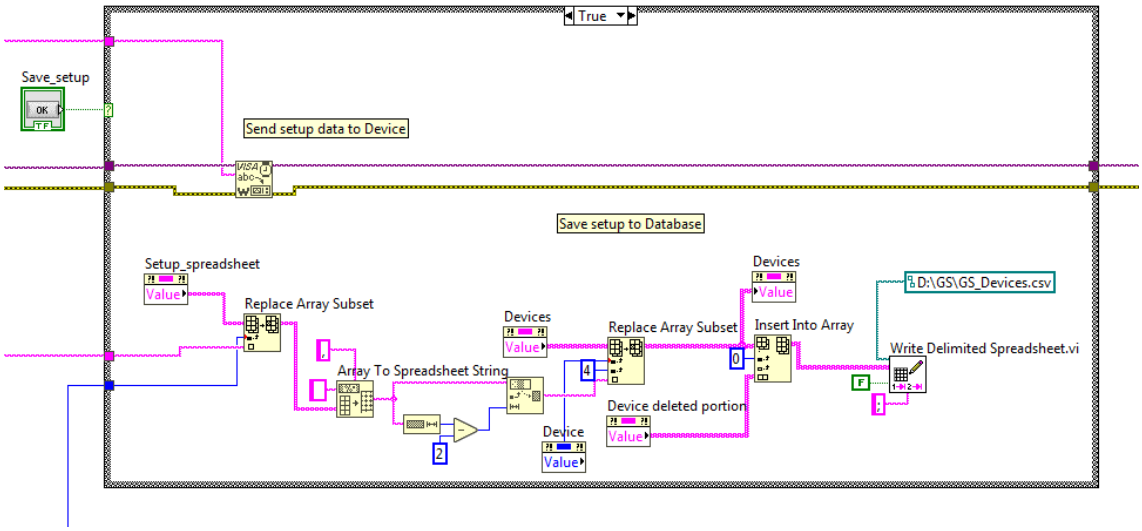



Рисунок 4.9 – Надсилання повідомлень через COM порт

Програма GS-Dashboard відображає результати вимірювань параметрів та дозволяє користувачеві здійснювати моніторинг системи. Зовнішній вигляд інтерфейсу користувача програми показано на рис. 4.10.

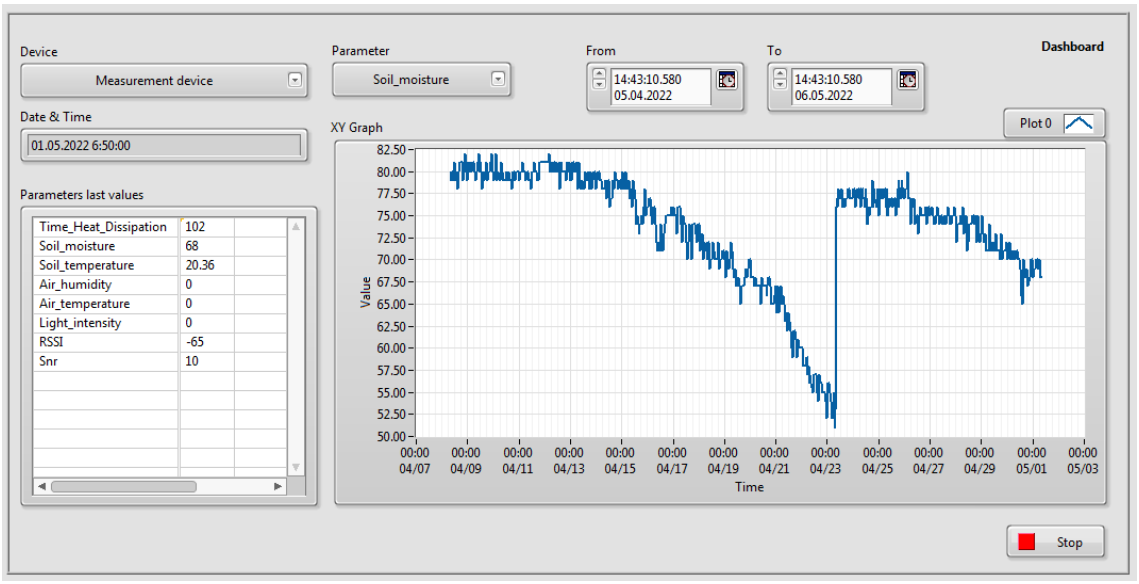



Рисунок 4.10 – Інтерфейс користувача програми GS-Dashboard

Селектор Device призначений для вибору пристрою. Селектори Parameter та From, To дозволяють вибирати параметр та діапазон часу для виведення даних на графік. Останнє поточне значення параметрів пристрою відображається у таблиці Parameters last values.

Під час запуску програми виконується ініціалізація змінних (рис. 4.11).

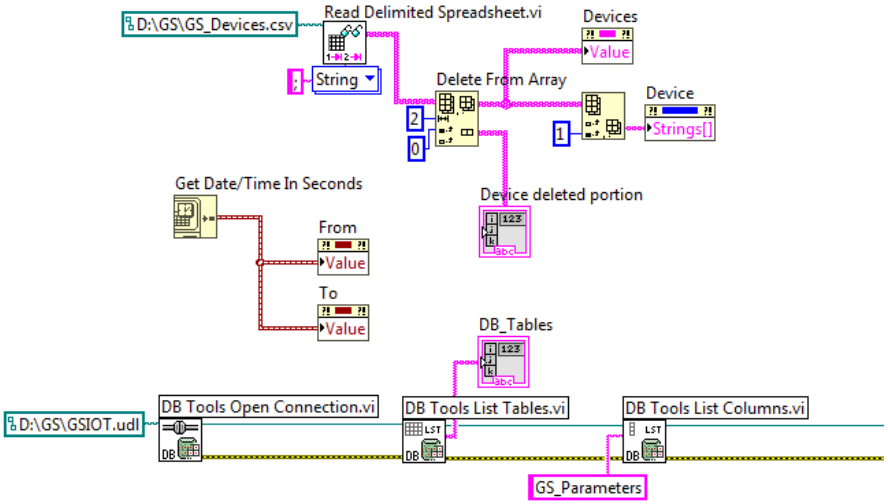



Рисунок 4.11 – Ініціалізація змінних програм GS-Dashboard

Фрагмент програми вибору пристрою показано на рисунку 4.12.

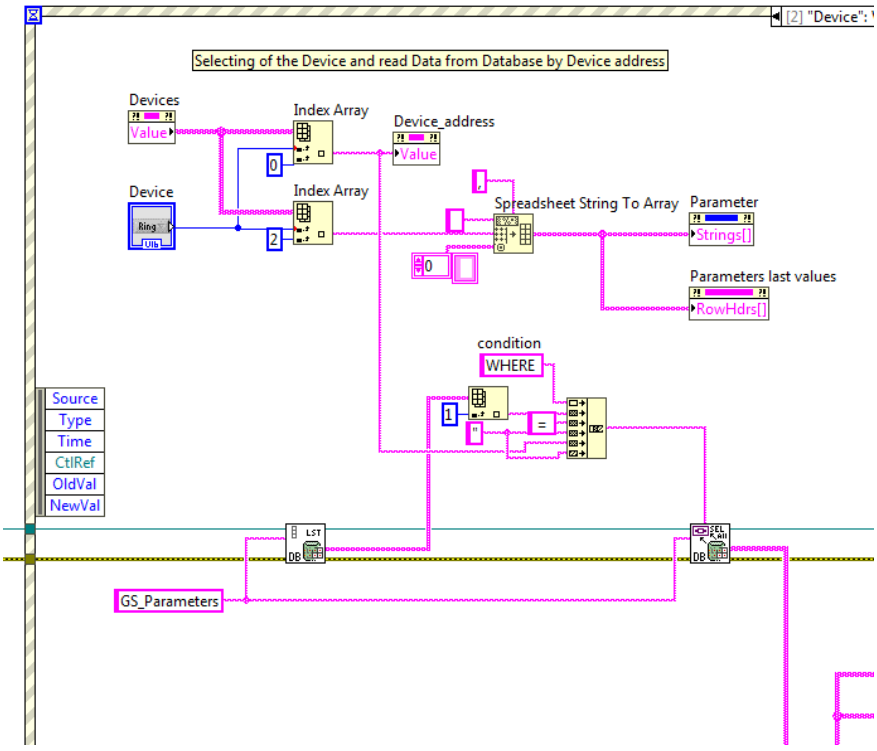



Рисунок 4.12 – Фрагмент програми вибору пристрою

На рисунку 4.13 показаний фрагмент програми виведення поточного значення параметрів пристрою.

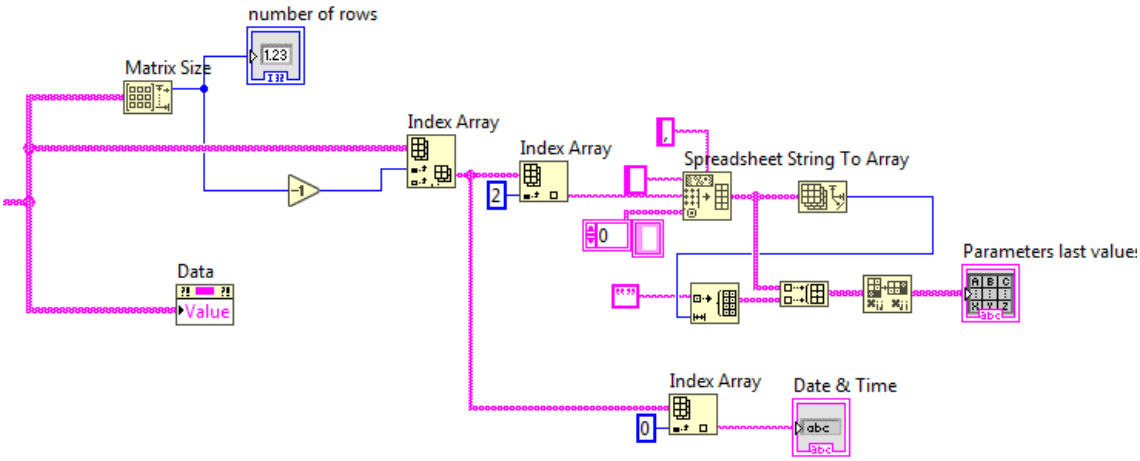



Рисунок 4.13 – фрагмент програми виведення поточного значення параметрів пристрою

Фрагмент програми для графічного відображення вибраного параметра пристрою показано на рисунку 4.14.

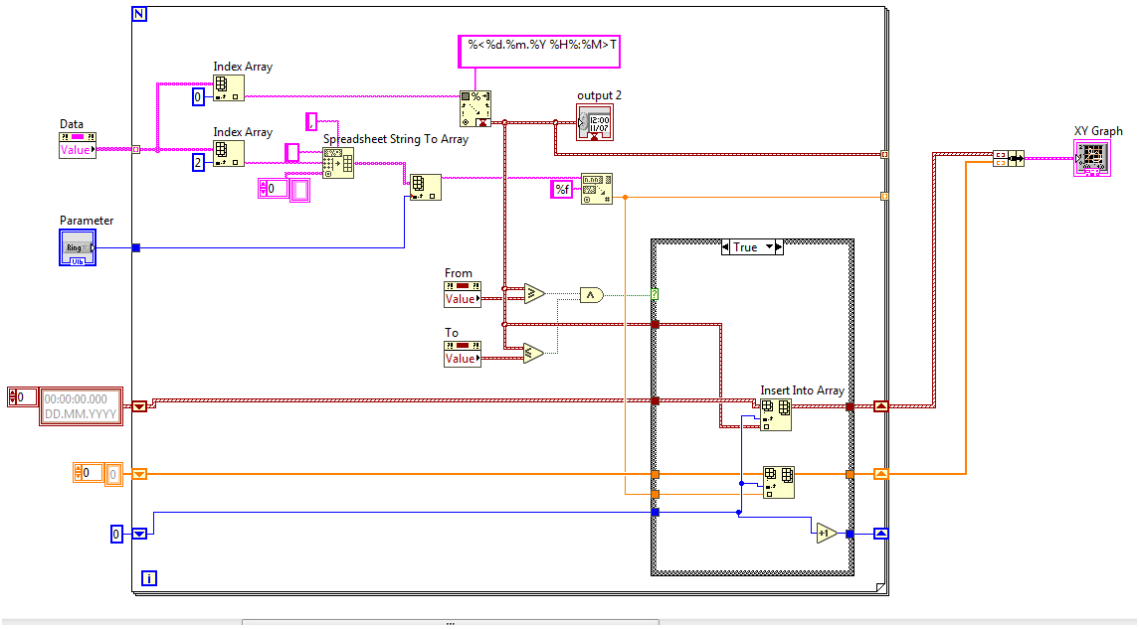



Рисунок 4.14 – Фрагмент програми для графічного відображення вибраного параметра пристрою

Інструкцію для користувача системи наведено в Розділі 6 .

# 5 АПРОБАЦІЯ СИСТЕМИ

## 5.1 Вибір сонячної панелі

Сонячні батареї, які також називають сонячними панелями або сонячними модулями, будуються з окремих фотоелектричних перетворювачів (так званих сонячних елементів), які з'єднуються один з одним у послідовні та паралельні ланцюги, що працюють у сукупності як єдине джерело струму.

Сьогодні реальний ККД сонячних батарей, доступних широкому споживачеві, лежить у межах від 17 до 23%. Є окремі екземпляри, які декларують ККД до 24%, але це швидше винятки та перебільшення. Лабораторії по всьому світу прагнуть розробити сонячні елементи, ККД яких хоча б наблизився до 30% – це було б дуже добрим результатом для джерела енергії даного типу, якщо дивитися на речі реально.

Сонячні батареї на основі кремнію, як альтернативне джерело електричної енергії, перевірені часом, вони відрізняються надійністю та безпекою, компактністю та відносною доступністю. Термін їх нормальної експлуатації сягає 30 років і навіть перевищує. Хоча, заради справедливості варто відзначити, що кремнієві фотоелектричні елементи з часом деградують, це виявляється у зниженні потужності, що отримується при повному освітленні, приблизно на 10% від початкового номіналу за кожні 10 років активної експлуатації. Що стосується тонкоплівкових елементів, то вони часом не перевірені, але фахівці стверджують, що швидкість деградації протягом перших років експлуатації у них багаторазово вища, ніж у монокристалічних та полікристалічних кремнієвих елементів.

При нормальній експлуатації ні заміна елементів, ні будь-яке інше спеціальне обслуговування монокристалічним та полікристалічним сонячним панелям не потрібно. Вони прості в установці, не містять частин, що

рухаються, їх поверхня, звернена до Сонця, завжди має захисне механічно міцне покриття.

Вольт – амперна характеристика сонячних батарей знімається в лабораторних умовах при виробництві та наводиться у специфікації. Стандартний тест проводиться за сонячної радіації 1000 Вт/кв.м при температурі навколишнього повітря 25°С, як на широті 45°.

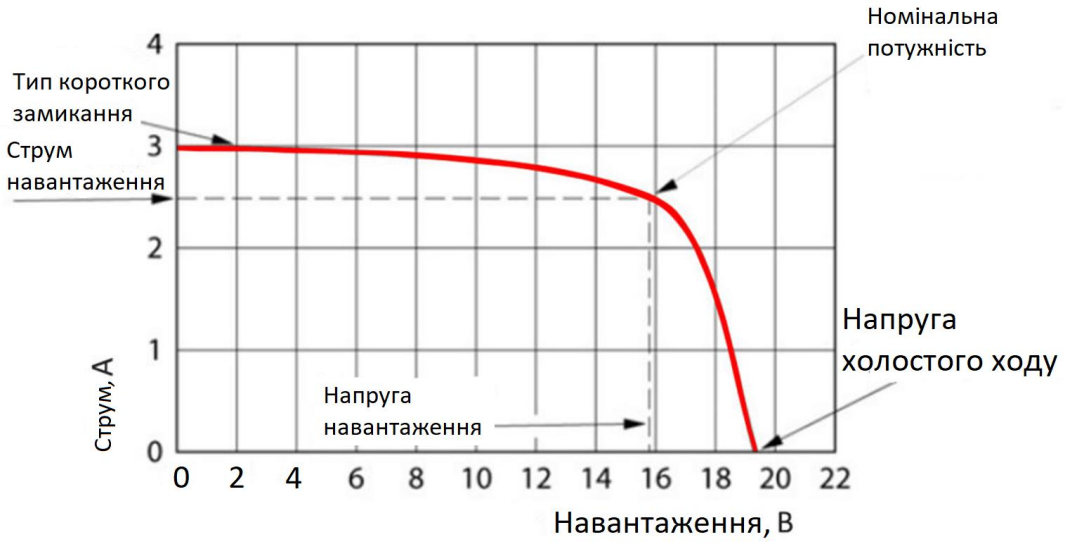



Рисунок 5.1 – Вольт-амперна характеристика сонячної панелі 40 Вт

Потужність сонячної батареї змінюється залежно від інтенсивності сонячної радіації і може коливатись як протягом доби у значних межах, а також залежно від місяця року.


Рисунок 5.2 – Денне генерування сонячної панелі потужність 1 кВт

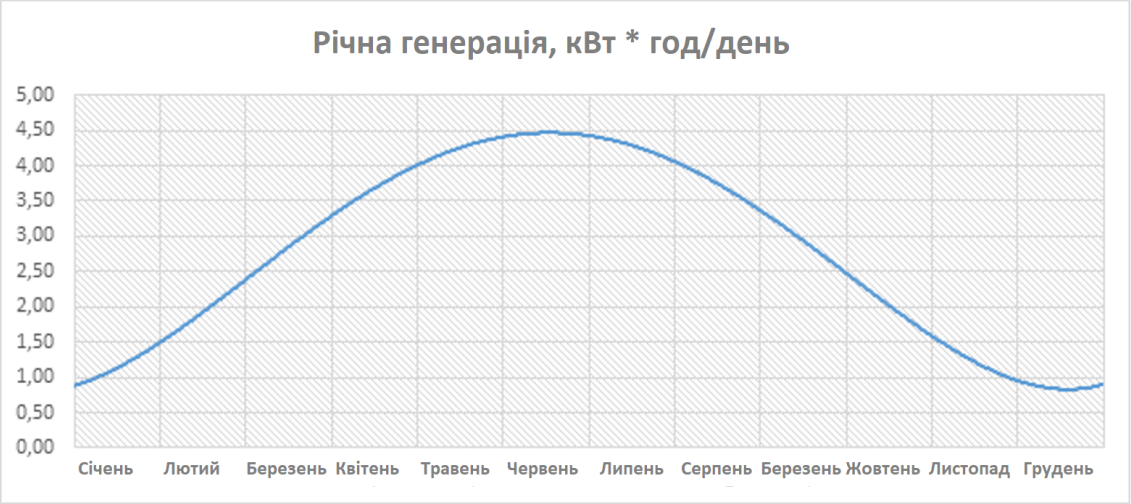



Рисунок 5.3 – Місячна генерація сонячної панелі 1 кВт

Електрична потужність, необхідна для живлення пристрою:

P=0.15A x 5.0B=0.75 Вт.

З урахуванням ККД перетворювача, необхідна потужність джерела струму має бути не менше 1 Вт.

З урахуванням вище викладеного та потужності, що споживається пристроєм, для забезпечення живлення підійде сонячна панель номінальної потужності 30 Вт розміром ***х*х* мм.

Сонячна панель потужністю 30 Вт має номінальні параметри: Струм навантаження 1.75А; напруга 17 Ст.

При необхідності передачі результатів вимірювання цілодобово сонячна панель може бути дообладнана блоком акумуляторів із зарядним пристроєм.

## 5.2 Конструкція для встановлення пристрою

Конструкція для встановлення пристрою повинна відповідати таким основним вимогам:

1) Забезпечувати зручне та надійне кріплення елементів пристрою;
2) Висота розташування сонячної панелі та антени не менше 2м;
3) Витримувати вплив довкілля (тепло, вітер, дощ, корозійна стійкість);
4) Забезпечувати зручність транспортування;
5) Мати гарний естетичний вигляд.

Основу конструкції складають:

1) Каркас для кріплення сонячної панелі;
2) Штанга заввишки 2 м;
3) Підстава для встановлення на землі;

На каркасі під сонячною панеллю кріпиться водонепроникний корпус пристрою розміром 145х92х40 мм. Антени та датчик інтенсивності світла кріпляться за допомогою кронштейна на верхньому рівні сонячної панелі.

Датчик температури та вологості повітря розташовується під тінню сонячної панелі та захищений від прямого попадання дощу.

Датчик вологості ґрунту встановлюється на необхідну глибину, залежно від виду рослин, на відстані 2-3 м від монтажної конструкції.

Досвідчений зразок системи було встановлено при присадибній ділянці щодо випробувань.

Рисунок 5.5 – Досвідчений зразок системи, встановлений на присадибній ділянці

На рисунку 5.6 та рисунку 5.7 показані графіки зміни вологості та температури ґрунту, отримані при випробуваннях системи.

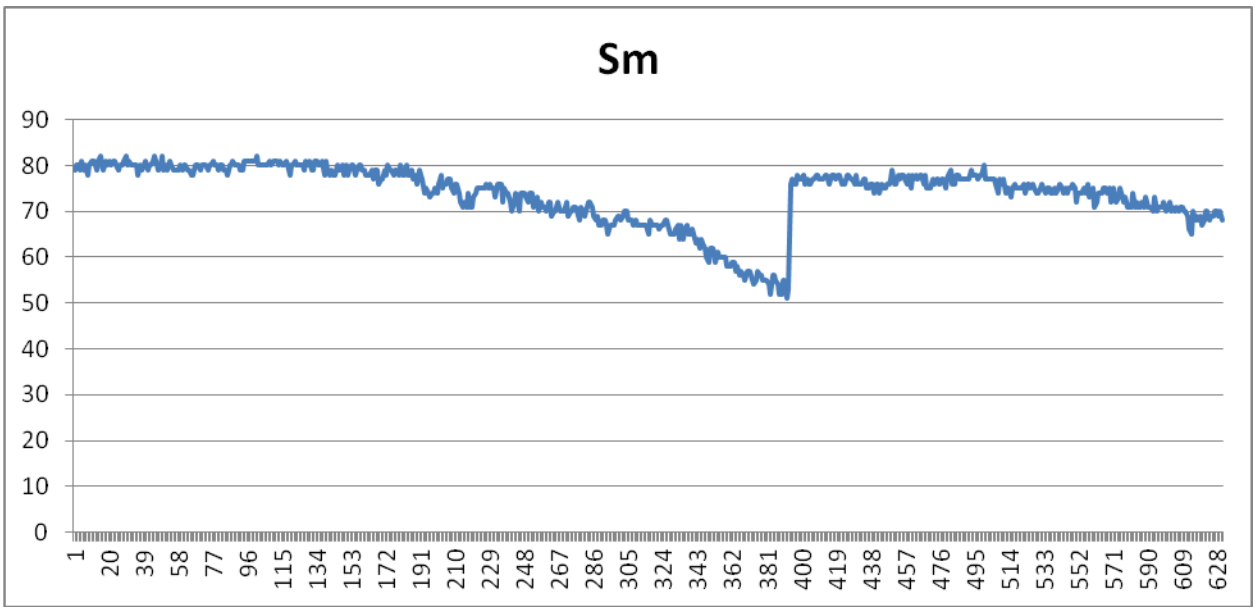



Рисунок 5.6 – Графік зміни вологості ґрунту

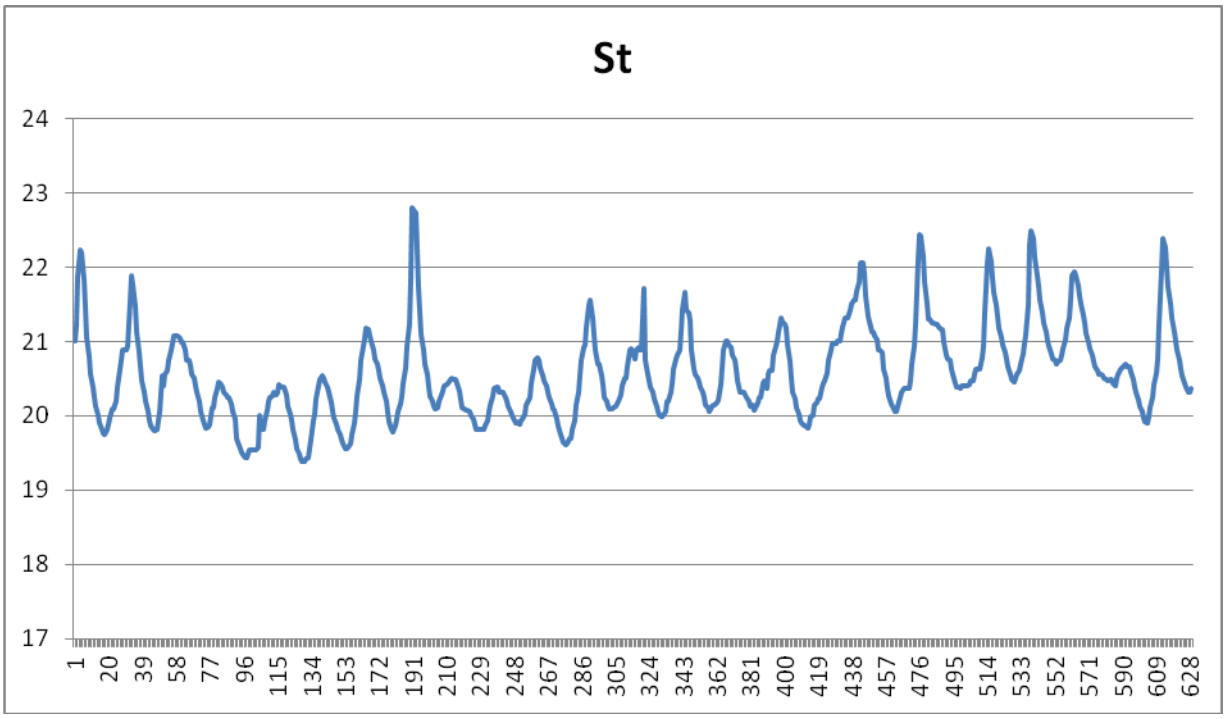



Рисунок 5.7 – Графік зміни температури ґрунту

Досвідчені випробування показали, що система виконує поставлені завдання.

# 6 ПОСІБНИК КОРИСТУВАЧА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення (ПЗ) призначене для моніторингу параметрів та налаштування компонентів системи управління зрошенням (СУО).

ПЗ працює під керуванням ОС Windows 7.0 та вище.

ПЗ включає такі основні файли:

5) Програму GS_Control;
6) Програму GS_Dashboard;
7) Базу даних GSIOT;
8) Файл конфігурації пристроїв GS_Devices.

Встановлення програмного забезпечення.

ПЗ поставляється на флеш-накопичувачі. Щоб інсталювати програмне забезпечення, запустіть файл setup.exe і дотримуйтесь інструкцій з інсталяції.

До складу СУО входять такі основні типи пристроїв:

1) ПЗП - пристрій вимірювання та передачі параметрів;
2) ПКЗ - пристрій управління зрошенням;
3) ШПД – шлюз передачі даних.

Кількість ПЗП та ПКЗ визначається кількістю ділянок зрошення. ШПД підключається до ПК, де встановлено справжнє ПЗ. Обмін даних між компонентами системи відбувається бездротовим каналом передачі LoRa на частоті 433 МГц.

Всі пристрої попередньо налаштовані та дозволяють організувати роботу системи одразу після монтажу компонентів на ділянках зрошення.

Файл конфігурації GS_Devices.csv містить інформацію про параметри налаштування компонентів системи.

До параметрів налаштування ПЗП належать:

1) Ім'я пристрою
2) Адреса системи
3) Адреса пристрою
4) Показання датчика вологості для сухого ґрунту.

5) Показання датчика вологості для мокрого ґрунту.

Перші три параметри задаються виробником під час постачання системи. Параметри 4 та 5 користувач може змінити залежно від умов використання системи.

До параметрів налаштування ПЗВ відносяться:

1) Найменування пристрою;
2) Адреса системи;
3) Адреса пристрою;
4) Адреса ПЗП;
5) Показання ПЗП у якому включається зрошення;
6) Показання ПЗП у якому зрошення вимикається;
7) час включення зрошення;
8) Час вимкнення зрошення.

Перші 4 параметри задаються виробником під час постачання системи. Параметри 5-8 користувач може змінити залежно від умов використання системи.

Після встановлення пристроїв на ділянках запустіть програму GS_Control. Програма забезпечує інтерфейс із ШПД та можливість налаштування параметрів пристроїв системи. Зовнішній вигляд інтерфейсу користувача показано на рисунку 6.1.

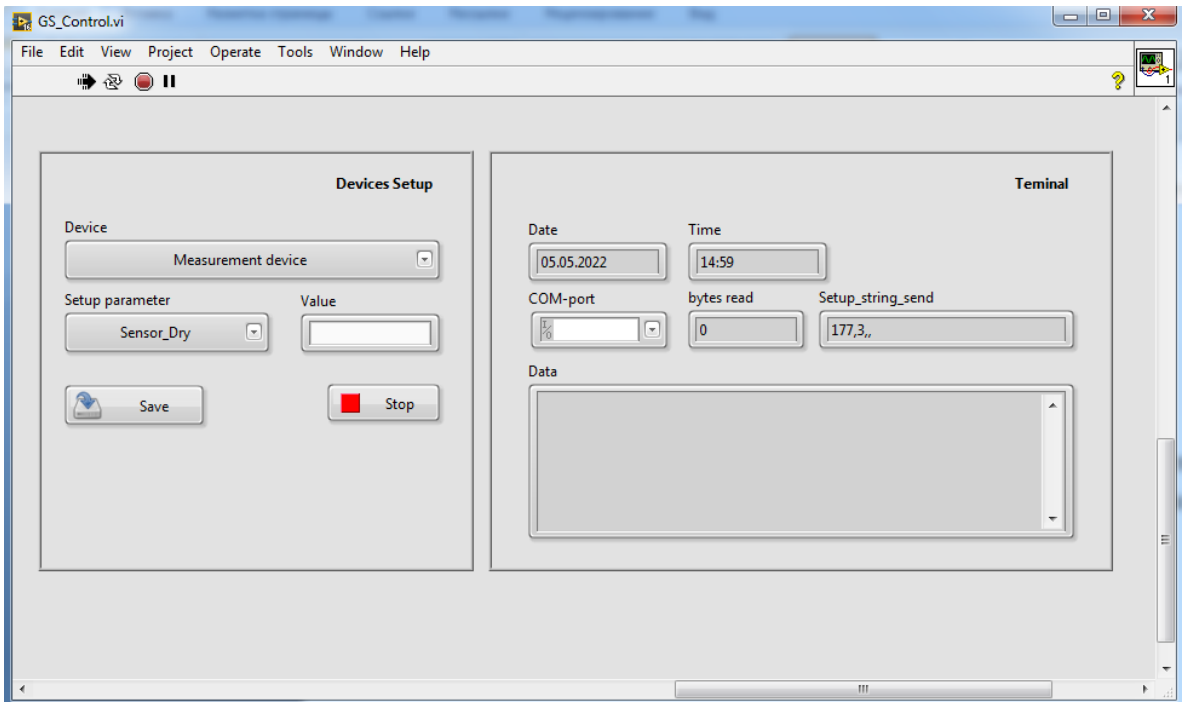



Рисунок 6.1 – Інтерфейс користувача програми GS_Control

Ліворуч розташована область Device Setup, призначена для виконання операцій з настроювання параметрів пристроїв системи. Селектор Device забезпечує вибір пристрою. Селектор Setup parameter призначений для вибору параметра. У полі Value вводиться нове значення параметра. За допомогою кнопки Save нове значення параметра записується у файл конфігурації та передається пристрою через ШПД.

Процес обміну даними з пристроями відображається в області Terminal.

Програма GS-Dashboard відображає результати вимірювань параметрів та дозволяє користувачеві здійснювати моніторинг системи. Зовнішній вигляд інтерфейсу користувача програми показано на рисунку 6.2.

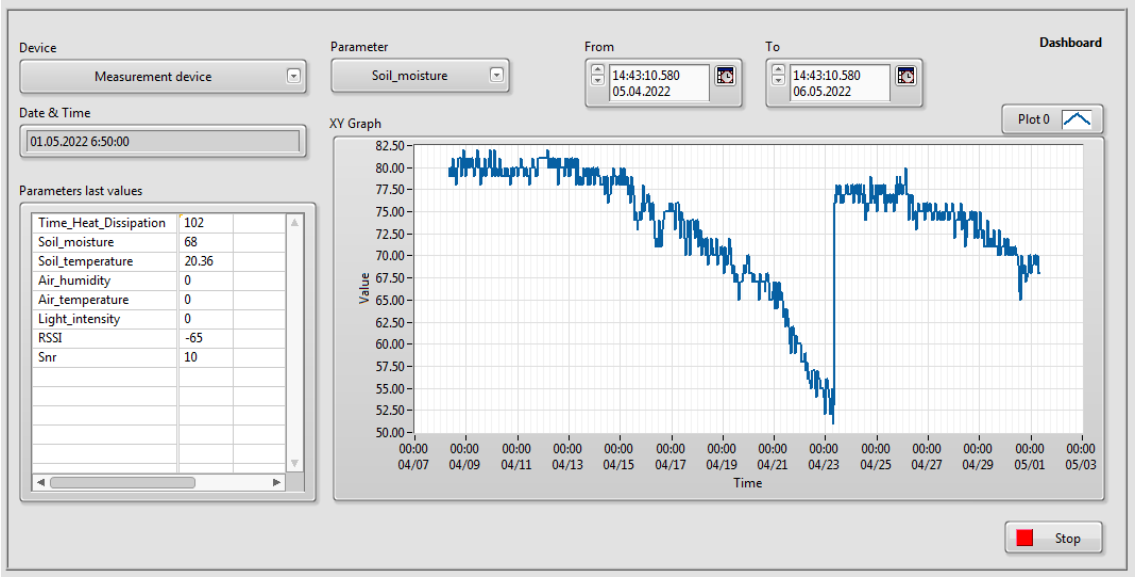



Рисунок 6.2 – Інтерфейс користувача програми GS-Dashboard

Селектор Device призначений для вибору пристрою. Селектори Parameter та From, To дозволяють вибирати параметр та діапазон часу для виведення даних на графік. Останнє поточне значення параметрів пристрою відображається у таблиці Parameters last values.

# ВИСНОВОК

В результаті виконаної роботи була проаналізована предметна область, сформовано перелік вимог до розроблямої системи. Окремо було досліджено сучасний ринок апаратних компонентів, придатних для використання в проекті. На базі цього дослідження було обрано необхідні модулі, на базі яких створено прототипи пристроїв:

- пристрій вимірювання та передачі параметрів (ПВП);
- пристрій управління періодичністю та часом зрошення на основі показань (ПВП);
- шлюз для передачі даних.

Для кожного пристрою було розроблено та протестовано програмне забезпечення, за допомогою якого можно дистанційно керувати системою та здійснювати моніторинг параметрів.

Були проведені дослідження та апробація системи автоматичного керування зрошенням ґрунту з автономним живленням в польових умовах.

Дана система може використовуватись сільськогосподарськими підприємствами.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Soil Moisture & Temperature Sensor, 2021. – Режим доступу: https://github.com/greensensors/SMT-Soil-Moisture-Sensor-for-Arduino
2. Soil Moisture and Temperature sensor SMT-01, 2022. – Режим доступу: https://hackaday.io/project/177850-soil-moisture-and-temperature-sensor-smt-01
3. Procedure for calculating reference and crop evapotranspiration from meteorological data and crop coefficients, 1998. – Режим доступу: https://www.fao.org/3/x0490e/x0490e00.htm#Contents
4. Методи вимірювання вологості ґрунту, 2018. – Режим доступу: https://propozitsiya.com/metody-izmereniya-vlazhnosti-pochvy
5. Дощувальні машини для поливу полів, 2022. – Режим доступу: https://nrg-group.ua/
6. Іригаційні рішення, 2019. – Режим доступу: https://fregat.mk.ua/ru/produksiya/irrigaczionnye-resheniya/
7. Introducing the irrigation installation fundamental program, 2022. – Режим доступу: https://www.hunterindustries.com/
8. Drip Irrigation, 2022. – Режим доступу: https://www.rainbird.com/
9. Irrigation Systems, 2006. – Режим доступу: https://www.irritrol.it/eng/
10. METER Environment Knowledge Base, 2021. – Режим доступу: https://www.metergroup.com/en/meter-environment/knowledge-base

# ДОДАТОК А

Характеристики датчику для вимірювання температури та вологості повітря HTU20D наведено на рисунку А.1 та рисунку А.2:

| Ratings | Symbol | Value | Unit |
|---|---|---|---|
| Storage Temperature | $T_{stg}$ | -40 to 125 | °C |
| Supply Voltage (Peak) | $V_{cc}$ | 3.8V | $V_{dc}$ |
| Humidity Operating Range | RH | 0 to 100 | %RH |
| Temperature Operating Range | $T_a$ | -40 to +125 | °C |
| VDD to GND | | -0.3 to 3.6V | V |
| Digital I/O pins (DATA/SCK) to VDD | | -0.3 to VDD+0.3 | V |
| Input current on any pin | | -10 to +10 | mA |

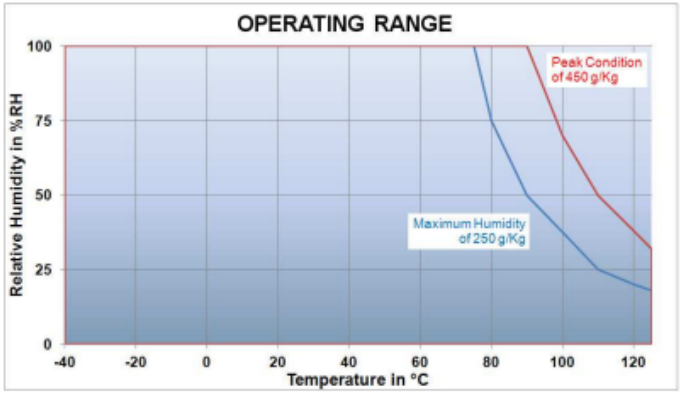
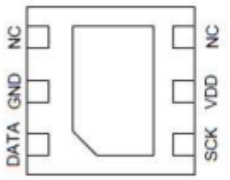



Рисунок А.1 – Технічні характеристики датчика HTU20D

| N° | Function | Comment |
|---|---|---|
| 1 | DATA | Data bit-stream |
| 2 | GND | Ground |
| 3 | NC | Must be left unconnected |
| 4 | NC | Must be left unconnected |
| 5 | VDD | Supply Voltage |
| 6 | SCK | Selector for RH or Temp |
| PAD | | Ground or unconnected |

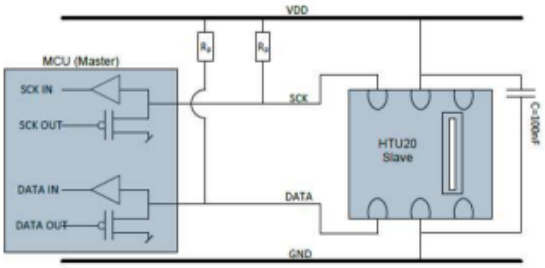
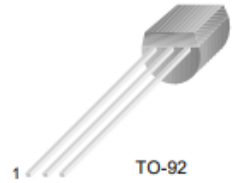




Рисунок А.2 – Специфікація інтерфейсу

На рисунку А.3 показано характеристики транзистора 2N2222:

**PN2222**

**General Purpose Transistor**



1 TO-92

1. Emitter 2. Base 3. Collector

**NPN Epitaxial Silicon Transistor**

**Absolute Maximum Ratings** $T_a$=25°C unless otherwise noted

| Symbol | Parameter | Value | Units |
|---|---|---|---|
| $V_{CBO}$ | Collector-Base Voltage | 60 | V |
| $V_{CEO}$ | Collector-Emitter Voltage | 30 | V |
| $V_{EBO}$ | Emitter-Base Voltage | 5 | V |
| $I_C$ | Collector Current | 600 | mA |
| $P_C$ | Collector Power Dissipation | 625 | mW |
| $T_J$ | Junction Temperature | 150 | °C |
| $T_{STG}$ | Storage Temperature | -55 ~ 150 | °C |

Рисунок А.3 – Технічні характеристики та контакти транзистора 2N2222

Характеристики фоторезистору GL55 показано на рисунку А.4:

| Specification | Type | Max. Voltage | Max. power | Environmental temp. | Spectrum peak value |
|---|---|---|---|---|---|
| Φ5 series | GL5516 | 150 | 90 | -30~+70 | 540 |
| | GL5528 | 150 | 100 | -30~+70 | 540 |
| | GL5537-1 | 150 | 100 | -30~+70 | 540 |
| | GL5537-2 | 150 | 100 | -30~+70 | 540 |
| | GL5539 | 150 | 100 | -30~+70 | 540 |
| | GL5549 | 150 | 100 | -30~+70 | 540 |

| Specification | Light resistance (10Lux) (KΩ) | Dark resistance (MΩ) | $\gamma_{10}^{100}$ | Response time (ms) | | Illuminance resistance Fig. No. |
|---|---|---|---|---|---|---|
| | | | | Increase | Decrease | |
| Φ5 series | 5-10 | 0.5 | 0.5 | 30 | 30 | 2 |
| | 10-20 | 1 | 0.6 | 20 | 30 | 3 |
| | 20-30 | 2 | 0.6 | 20 | 30 | 4 |
| | 30-50 | 3 | 0.7 | 20 | 30 | 4 |
| | 50-100 | 5 | 0.8 | 20 | 30 | 5 |
| | 100-200 | 10 | 0.9 | 20 | 30 | 6 |

Рисунок А.4 – Технічні характеристики фоторезистора GL55

На рисунку А.5 показано характеристики цифрового термометру DS18B20:

| Specification | Type | Max. Voltage | Max. power | Environmental temp. | Spectrum peak value |
|---|---|---|---|---|---|
| Φ5 series | GL5516 | 150 | 90 | -30~+70 | 540 |
| | GL5528 | 150 | 100 | -30~+70 | 540 |
| | GL5537-1 | 150 | 100 | -30~+70 | 540 |
| | GL5537-2 | 150 | 100 | -30~+70 | 540 |
| | GL5539 | 150 | 100 | -30~+70 | 540 |
| | GL5549 | 150 | 100 | -30~+70 | 540 |

| Specification | Light resistance (10Lux) (KΩ) | Dark resistance (MΩ) | $\gamma_{10}^{100}$ | Response time (ms) | | Illuminance resistance Fig. No. |
|---|---|---|---|---|---|---|
| | | | | Increase | Decrease | |
| Φ5 series | 5-10 | 0.5 | 0.5 | 30 | 30 | 2 |
| | 10-20 | 1 | 0.6 | 20 | 30 | 3 |
| | 20-30 | 2 | 0.6 | 20 | 30 | 4 |
| | 30-50 | 3 | 0.7 | 20 | 30 | 4 |
| | 50-100 | 5 | 0.8 | 20 | 30 | 5 |
| | 100-200 | 10 | 0.9 | 20 | 30 | 6 |

Рисунок А.5 – Технічні характеристики цифрового термометра DS18B20

Характеристики приймач-передавач модуляю Ai-Thinker Ra-02 LoRa наведена на рисунку А.6:

## Product Specifications

| | |
|---|---|
| Module Model | Ra-02 |
| Package | SMD-16 |
| Size | 17*16*(3.2 ± 0.1) mm |
| Interface | SPI |
| Programmable bit rate | UP to 300Kbps |
| Frequency Range | 410-525 MHz |
| Antenna | IPEX |
| Max Transmit Power | 18±1 dBm |
| Power (Typical Values) | 433MHz:<br>TX:93mA RX:12.15mA Standby:1.6mA<br>470MHZ:<br>TX:97mA RX:12.15mA Standby:1.5mA |
| Power Supply | 2.5~3.7V, Typical 3.3V |
| Operating Temperature | -30 ℃ ~ 85 ℃ |
| Storage Environment | -40 ℃ ~ 90 ℃ , < 90%RH |
| Weight | 0.45g |

## Receive Sensitivity

| Frequency | Spread Factor | SNR | Sensitivity |
|---|---|---|---|
| 433MHz | 7 | -7 | -125 |
| | 10 | -15 | -134 |
| | 12 | -20 | -141 |
| 470MHz | 7 | -7 | -126 |
| | 10 | -15 | -135 |
| | 12 | -20 | -141 |

Рисунок А.6 – Технічні характеристики модуля Ai-Thinker Ra-02 LoRa

На рисунку А.7 показані технічні характеристика МК Arduino Nano:

**Summary**

| | |
|---|---|
| Microcontroller | Atmel ATmega328 |
| Operating Voltage (logic level) | 5 V |
| Input Voltage (recommended) | 7-12 V |
| Input Voltage (limits) | 6-20 V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 8 |
| DC Current per I/O Pin | 40 mA |
| Flash Memory | 16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader |
| SRAM | 1 KB (ATmega168) or 2 KB (ATmega328) |
| EEPROM | 512 bytes (ATmega168) or 1 KB (ATmega328) |
| Clock Speed | 16 MHz |
| Dimensions | 0.73" x 1.70" |

**the board**

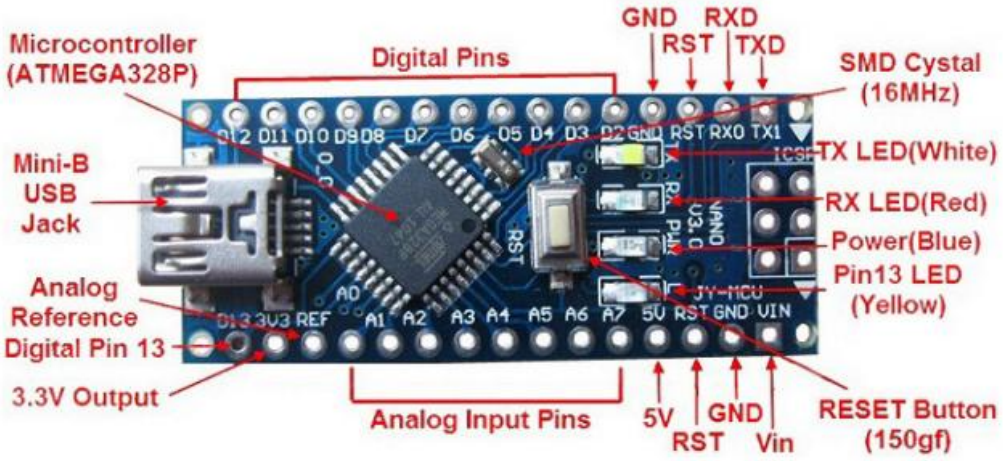



Рисунок А.7 – Технічні характеристики МК Arduino Nano

# ДОДАТОК Б

Програма для вимірюваняя температури та вологості ґрунту:

```
Soil Moisture & Temperature Sensor SMT-01
Example for Arduino UNO
SMT-01 use Heat Dissipation Method
Components of SMT-01:
DS18B20 - 1-Wire temperature sensor
2N2222A - as Heater
Connection wires of SMT-01 cable to Arduino UNO (see Electrical Circuit):
Yellow DS  - to Pin 5 (R2 4.7k - 2.0k to +5V, depending on length of the cable)
Green 2N   - to Pin 4 (via R1 (10k - 2.0k, depending on length of the cable)
Red        - to +5V
Black      - to GND

#include <OneWire.h>
#define DARK  LOW
#define LIGHT HIGH
#define ON    HIGH
#define OFF   LOW
OneWire  ds(5);  // 1-Wire to Pin 5
byte i;
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
float celsius;
int m_err;

int pin_Led = 13;
int pin_Heater = 4;
int DS_found = 0;
float         Time_Heat_Dissipation,         Soil_Moisture, Soil_Temperature;
unsigned long Heating_Time = 60000; //ms
int j, k;

void DS18B20_init(void);
void DS18B20_measure(void);
void Measure_SMT (void);
unsigned long mtime;
unsigned long set_mtime;

void setup()
{
   pinMode(pin_Led,OUTPUT);
   pinMode(pin_Heater, OUTPUT);
   digitalWrite(pin_Led,DARK);
```

```
        digitalWrite(pin_Heater, LOW);
    // UART communication setup
        Serial.begin(9600);
        delay(10);
        Serial.println("");
        Serial.println("Initialization of DS18B20 ... ");

        DS_found = 0;
        DS18B20_init();
        if (DS_found == 1){
            digitalWrite(pin_Led, LIGHT);
            Serial.println("Initialization is Ok");
            delay(1000);
            digitalWrite(pin_Led,DARK);
           }
        set_mtime = 1000;
        mtime = millis();
    }//setup

    void loop()
    {
     if (millis() - mtime > set_mtime) {
      digitalWrite(pin_Led, LIGHT);

    // Measurement
      if(DS_found == 1){
      Serial.println("Start measurement");
      Measure_SMT();

    // Converting the Time of Heat Dissipation to Soil Moisture,
%
    // as example
      float Sensor_Dry = 250.0; //Time of Heat Dissipation for
Dry Sensor
      float Sensor_Wet = 46.0;  //Time of Heat Dissipation for
Wet Sensor
      Soil_Moisture  =  map(Time_Heat_Dissipation,  Sensor_Dry,
Sensor_Wet, 0.0, 100.0);
      if (Soil_Moisture < 0.0) Soil_Moisture = 0.0;
      if (Soil_Moisture > 100.0) Soil_Moisture = 100.0;

      Serial.print("Soil Moisture = ");
      Serial.print(Soil_Moisture);
      Serial.println(", %");

      Serial.print("Temperature of Soil = ");
      Serial.print(Soil_Temperature);
      Serial.println(", oC");
      }
      else{
       Serial.println("Next try to Initialization of DS18B20 ...
");
       DS_found = 0;
```

```
         DS18B20_init();
         if (DS_found == 1){
             digitalWrite(pin_Led, LIGHT);
             Serial.println("Initialization is Ok");
             delay(1000);
             digitalWrite(pin_Led,DARK);
            }
           }

       set_mtime  =  420000;  //  recommended  pause  between
measurements, ms (7 minutes)
       mtime = millis();

       Serial.println("Waiting for the next measurement ...");
       digitalWrite(pin_Led,DARK);
       }// if mtime
       delay(10);
     }// loop

     void DS18B20_init(void){
     if ( !ds.search(addr)) {
         DS_found = 0;
         Serial.println("Sensor not found.");
         Serial.println();
         ds.reset_search();
         delay(250);
         return;
       }//if

     Serial.print("ROM =");
       for( i = 0; i < 8; i++) {
         Serial.write(' ');
         Serial.print(addr[i], HEX);
       }

       if (OneWire::crc8(addr, 7) != addr[7]) {
           Serial.println("CRC is not valid!");
           DS_found = 0;
           return;
       }
       Serial.println();

     // the first ROM byte indicates which chip
       switch (addr[0]) {
         case 0x10:
           Serial.println("  Chip = DS18S20");  // or old DS1820
           type_s = 1;
           break;
         case 0x28:
           Serial.println("  Chip = DS18B20");
           type_s = 0;
           DS_found = 1;
           break;
```

```
        default:
          Serial.println("Device is not a DS18x20 family
device.");
          return;
       }//switch
     }//DS18B20_init

     void DS18B20_measure(void)
     {
       m_err = 0;
       ds.reset();
       ds.select(addr);
       ds.write(0x44, 1);     // start conversion, with parasite
power on at the end
       delay(1000);          // time need for conversion
       present = ds.reset();
       ds.select(addr);
       ds.write(0xBE);       // Read Scratchpad

       //Serial.print("  Data = ");
       //Serial.print(present, HEX);
       //Serial.print(" ");
       for ( i = 0; i < 12; i++) {           // we need 12 bytes
resolution
         data[i] = ds.read();
         //Serial.print(data[i], HEX);
         //Serial.print(" ");
       }
       //Serial.print(" CRC=");
       //Serial.print(OneWire::crc8(data, 8), HEX);
       //Serial.println();

      if (OneWire::crc8(data, 8) != data[8]) {
          Serial.println("CRC is not valid!");
          m_err = 1;
       }

       // Convert the data to actual temperature
       int16_t raw = (data[1] << 8) | data[0];
       if (type_s) {
         raw = raw << 3; // 9 bit resolution default
         if (data[7] == 0x10) {
           // "count remain" gives full 12 bit resolution
           raw = (raw & 0xFFF0) + 12 - data[6];
         }
       } else {
         byte cfg = (data[4] & 0x60);
         // at lower res, the low bits are undefined, so let's
zero them
         if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution,
93.75 ms
         else if (cfg == 0x20) raw = raw & ~3; // 10 bit res,
187.5 ms
```

```
        else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375
ms
        //// default is 12 bit resolution, 750 ms conversion
time
      }
       celsius = (float)raw / 16.0;
     }//DS18B20_measure

     void Measure_SMT () {
         float t_current, tj;
         int m_cycle, j, m;
         unsigned long dtime;
         Time_Heat_Dissipation = 0.0;
         j = 0;
         m = 0;
         tj = 0.0;

         Serial.println("Temperature of Soil measurement ...");
         for (m_cycle = 0; m_cycle<10; m_cycle++){
               DS18B20_measure();
               if (m_err == 0)
                  {
                   tj = tj + celsius;
                   j++;
                   Serial.println(celsius);
                  }//if
               else {m++;}
              }//for

         if (j > 0 && m < 7)  { Soil_Temperature = tj/j; }
         else {Soil_Temperature = -21.0;}
         if (Soil_Temperature > 0.0) {

         Serial.println("Heating ...");
         digitalWrite(pin_Heater, HIGH);
         delay(Heating_Time);                          // Time of
heating, ms
         digitalWrite(pin_Heater, LOW);
         Serial.println("Heat dissipation ...");
         dtime = millis(); // start time of Heat dissipation
         t_current = (Soil_Temperature + 5.0);
         DS18B20_measure();
         if (m_err == 0) {t_current = celsius;}
         m_cycle = 0;

         while (t_current > (Soil_Temperature + 1.0) && m_cycle <
250){
              DS18B20_measure();
              if (m_err == 0) {t_current = celsius;}
              Serial.println(t_current );
              m_cycle++;
             }//while
```

```
        Time_Heat_Dissipation = (millis() - dtime)/1000.0;
        Serial.print("Time of Heat Dissipation = ");
        Serial.print(Time_Heat_Dissipation);
        Serial.println(", seconds");
       }//Soil_Temperature > 0
    }// Measure_SMT
```

Програма для ПВП

```
    /*
      LoRa Half-Duplex communication via AI-Thiker Ra-2 LoRa
      MC: Arduino Nano
      SMT-01 - Soil Moisture & Temperature Sensor
      HTU21D - Air temperature and humidity Sensor (R-Vc, B-GND,
Ж-DA, З-CL)
    */

    #include <SPI.h>              // include libraries
    #include <LoRa.h>
    #include <EEPROM.h>
    #include <OneWire.h>
    #include "SparkFunHTU21D.h"
    #define DARK  LOW
    #define LIGHT HIGH

    OneWire  ds(5);               // 1-Wire to GPIO5
    HTU21D myHumidity;                  // Датчик температуры и
влажности HTU21D

    const long frequency = 433E6; // LoRa Frequency
    const int  csPin = 10;        // LoRa radio chip select
    const int  resetPin = 9;      // LoRa radio reset
    const int  irqPin = 2;        // change for your board; must
be a hardware interrupt pin

    String outgoing;              // outgoing message
    String message = "";
    byte systemID = 0xA1;         // ID irrigation system
    byte sensorID = 0xB1;         // address of this device
    byte messageID = 0xFF;        // address of destination
    byte controlID = 0x11;        // ID control device

    // EEPROM coefficints
    int smem = 512;
    float kx = 0.0;
    int l = 6;
    char floatbufVar[7];
    int Setup_data;
    byte i;
    byte present = 0;
    byte type_s;
    byte data[12];
    byte addr[8];
```

```
     float celsius;
     int m_err;
     int j, k, h, t;
     long zi, zp;
     int pin_Led = 13;
     int pin_Heater = 4;
     int DS_found = 0;                        // Initialization of
DS18B20 result
     float Time_Heat_Dissipation;
     float    Air_humidity,    Air_temperature,    Soil_moisture,
Soil_temperature, Light_int;
     unsigned long Heating_Time = 60000; //ms
     float  Sensor_Dry  =  250.0;                  //Time  of  Heat
Dissipation for Dry Sensor
     float  Sensor_Wet  =  35.0;                   //Time  of  Heat
Dissipation for Wet Sensor

     // Timing coefficients
     unsigned long mtime;
     unsigned  long  set_mtime;                // Measurement Time
period, ms
     float  mtimesetup  =  5.0;                // Measurement Time
period, minutes
     int sensorPin = A1;                   // select the input pin
for the potentiometer
     int sensorValue = 0;                 // variable to store the
value coming from the sensor

     void ReadFromMem(void);
     void WriteToMem(void);
     void DS18B20_init(void);
     void DS18B20_measure(void);
     void Measure_SMT (void);
     void(* resetFunc) (void) = 0;       //Reset with address 0

     // Time for Reset
     unsigned long resettime;
     unsigned long set_resettime = 86400000;     // Reset MC Time
period, ms (one/day)

     void setup() {
       set_mtime = 5000;          // 1-st measurement will start
after  starting,  the  next  measurements  -  every  "mtimesetup"
minutes
       mtime      = millis();
       resettime  = millis();
       pinMode(pin_Led,OUTPUT);
       pinMode(pin_Heater, OUTPUT);
       digitalWrite(pin_Led,DARK);
       digitalWrite(pin_Heater, LOW);
       Serial.begin(9600);                    // initialize serial
       // Reading setup coefficient from EEPROM
       //WriteToMem();    // need for first time
```

```
        ReadFromMem();
        Serial.println("Initialization of DS18B20 ... ");

        DS_found = 0;
        DS18B20_init();
        if (DS_found == 1){
             digitalWrite(pin_Led, LIGHT);
             Serial.println("Initialization is Ok");
             delay(1000);
             digitalWrite(pin_Led,DARK);
            }
        set_mtime = 1000;
        mtime = millis();
        myHumidity.begin();

        while (!Serial);
        Serial.println("LoRa Half Duplex");

        // override the default CS, reset, and IRQ pins (optional)
        LoRa.setPins(csPin, resetPin, irqPin);// set CS, reset,
IRQ pin
        if (!LoRa.begin(433E6)) {             // initialize ratio
at 433 MHz
          Serial.println("LoRa    init    failed.    Check    your
connections.");
          while (true);                           // if failed, do
nothing
        }//if lora
        Serial.println("LoRa init succeeded.");
      }// setup

      void loop() {

        // measure and sending
        if (millis() - mtime > set_mtime) {
        digitalWrite(pin_Led, LIGHT);
        if(DS_found == 1){
        Serial.println("Start measurement");
        Measure_SMT();
      // Converting the Time of Heat Dissipation to Soil Moisture,
%
        Soil_moisture  =  map(Time_Heat_Dissipation,  Sensor_Dry,
Sensor_Wet, 0.0, 100.0);
        if (Soil_moisture < 0.0) Soil_moisture = 0.0;
        if (Soil_moisture > 100.0) Soil_moisture = 100.0;

        Serial.print("Soil Moisture = ");
        Serial.print(Soil_moisture);
        Serial.println(", %");
        Serial.print("Temperature of Soil = ");
        Serial.print(Soil_temperature);
        Serial.println(", oC");
        }//DS is found
```

```
        else{
         Serial.println("Next try to Initialization of DS18B20 ...
");
         DS_found = 0;
         DS18B20_init();
         if (DS_found == 1){
             digitalWrite(pin_Led, LIGHT);
             Serial.println("Initialization is Ok");
             delay(1000);
             digitalWrite(pin_Led,DARK);
            }//if

          }//else

        delay (1000);
        float humd;
        float temp;
        h = 0;
        t = 0;
        for (i = 0; i<10; i++)
            {
             humd = myHumidity.readHumidity();
             temp = myHumidity.readTemperature();
             delay(1000);
             Serial.println(humd);
             Serial.println(temp);
             if (humd > 0.0 && humd < 100.0) {Air_humidity =
Air_humidity + humd; h++;}
             if (temp > 0.0 && temp < 81.0) {Air_temperature =
Air_temperature + temp; t++;}
           }//for
        if( h > 0) {Air_humidity = Air_humidity/h;} else
{Air_humidity = 0.0;}
        if( t > 0) {Air_temperature = Air_temperature/t;} else
{Air_temperature = 0.0;}

        Serial.print(" Air temperature: ");
        Serial.print(Air_temperature);
        Serial.print(",oC ");
        Serial.print(" Humidity: ");
        Serial.print(Air_humidity);
        Serial.print(",%");
        Serial.println();
        delay(1000);

        sensorValue = analogRead(sensorPin);
        Light_int = 5.0*((float)sensorValue)/1024.0;
         // Light_int = 0.0;

        message = "#";
        message+=Time_Heat_Dissipation;
        message+=",";
```

```
        message+=Soil_moisture;
        message+=",";
        message+=Soil_temperature;
        message+=",";
        message+=Air_humidity;
        message+=",";
        message+=Air_temperature;
        message+=",";
        message+=Light_int;
        message+="#";

        sendMessage(message);                  // send a message
        Serial.println("Sending " + message);
        set_mtime   =   420000;   //   recommended   pause   between
measurements, ms (7 minutes)
        mtime = millis();
        Serial.println("Waiting for the next measurement ...");
        digitalWrite(pin_Led,DARK);
        }//measure

        if (millis() - resettime > set_resettime) {
            // one per Day
            resetFunc();
           }//if mills
        // parse for a packet, and call onReceive with the result:
        onReceive(LoRa.parsePacket());
      }// loop

      void sendMessage(String outgoing) {
        LoRa.beginPacket();                    // start packet
        LoRa.write(systemID);                     // ID irrigation
system
        LoRa.write(sensorID);                   // address of this
device
        LoRa.write(messageID);                 // destination ID
        LoRa.write(outgoing.length());             //  add  payload
length
        LoRa.print(outgoing);                  // add payload
        LoRa.endPacket();                      // finish packet and
send it
      }//sendMessage

      void onReceive(int packetSize) {
        if (packetSize == 0) return;            // if there's no
packet, return
        // read packet header bytes:
        int systemADR = LoRa.read();           // systemADR address
        byte sender = LoRa.read();             // sender address
        byte incomingMsgId = LoRa.read();      // incoming msg ID
        byte  incomingLength  =  LoRa.read();       //  incoming  msg
length
        String incoming = "";
```

```
      while (LoRa.available()) {
        incoming += (char)LoRa.read();
      }

      if (incomingLength != incoming.length()) {    // check
length for error
        Serial.println("error: message length does not match
length");
        return;                                     // skip rest of
function
      }

      // if the systemADR isn't this device or broadcast,
      if (systemADR != systemID) {
        Serial.println("This message is not for me.");
        return;                                     // skip rest of
function
      }

     if (sender == controlID && incomingMsgId == sensorID) {

      Serial.println("Control command");
      Serial.println("System: 0x" + String(systemADR, HEX));
      Serial.println("Sender: 0x" + String(sender, HEX));
      Serial.println("Message ID: " + String(incomingMsgId));
      Serial.println("Message        length:        "        +
String(incomingLength));
      Serial.println("Message: " + incoming);
      Serial.println("RSSI: " + String(LoRa.packetRssi()));
      Serial.println("Snr: " + String(LoRa.packetSnr()));
      Serial.println();

      int cs = incoming.indexOf(",");
      String control_string = incoming.substring(0,cs);
      zi = control_string.toInt();
      Serial.println(zi);
      int fs = cs + 1;
      cs = incoming.indexOf(",", fs);
      control_string = incoming.substring(fs,cs);
      //float parameter = control_string.toFloat();
      Serial.println(control_string);

      if(zi    ==    1){    zp    =    control_string.toInt();
Serial.println(zp); systemID = (byte)zp;}
      if(zi    ==    2){    zp    =    control_string.toInt();
Serial.println(zp); sensorID = (byte)zp;}
      if(zi    ==    3){    kx    =    control_string.toFloat();
Serial.println(kx); Sensor_Dry = kx;}
      if(zi    ==    4){    kx    =    control_string.toFloat();
Serial.println(kx); Sensor_Wet = kx;}

      if(zi > 0 && zi <5){ WriteToMem(); }//if zi
      message = "";
```

```
        message+="!";
        message+=Sensor_Dry;
        message+=",";
        message+=Sensor_Wet;
        sendMessage(message);                    //confirmation -
Setup complete
        Serial.println("Sending " + message);
      }//if control
      }// receive

      void ReadFromMem(void){
           Serial.println("Reading coefficients from EPROM ... ");
           //EEPROM.begin(smem);
           j = l*0;
           for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
           kx = atof(floatbufVar);
           systemID = (byte)kx;
           Serial.println(systemID);

           j = l*1;
           for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
           kx = atof(floatbufVar);
           sensorID = (byte)kx;
           Serial.println(sensorID);

           j = l*2;
           for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
           kx = atof(floatbufVar);
           Sensor_Dry = kx;
           Serial.println(Sensor_Dry);

           j = l*3;
           for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
           kx = atof(floatbufVar);
           Sensor_Wet = kx;
           Serial.println(Sensor_Wet);

           //EEPROM.end();
           Serial.println("Completed");

      }// ReadFromMem()

      void WriteToMem(void){
        Serial.println("Writing data to EEPROM ... ");
        //EEPROM.begin(smem);
            j = l*0;
            kx = (float)systemID;
            dtostrf(kx, l, 1, floatbufVar);
```

```
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          j = l*1;
          kx = (float)sensorID;
          dtostrf(kx, l, 1, floatbufVar);
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          j = l*2;
          kx = Sensor_Dry;
          dtostrf(kx, l, 1, floatbufVar);
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          j = l*3;
          kx = Sensor_Wet;
          dtostrf(kx, l, 1, floatbufVar);
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          //EEPROM.commit();
          //EEPROM.end();
          Serial.println("Completed");
          Setup_data = 0;
     }//WriteToMem
     void DS18B20_init(void){ //-------------------------------
-----------------------------

     if ( !ds.search(addr)) {
         DS_found = 0;
         Serial.println("Sensor not found.");
         Serial.println();
         ds.reset_search();
         delay(250);
         return;
       }//if

     Serial.print("ROM =");
       for( i = 0; i < 8; i++) {
         Serial.write(' ');
         Serial.print(addr[i], HEX);
       }

       if (OneWire::crc8(addr, 7) != addr[7]) {
           Serial.println("CRC is not valid!");
           DS_found = 0;
           return;
       }
       Serial.println();

     // the first ROM byte indicates which chip
       switch (addr[0]) {
```

```
          case 0x10:
            Serial.println("  Chip = DS18S20");  // or old DS1820
            type_s = 1;
            break;
          case 0x28:
            Serial.println("  Chip = DS18B20");
            type_s = 0;
            DS_found = 1;
            break;
          default:
            Serial.println("Device   is   not   a   DS18x20   family
device.");
            return;
        }//switch

     }//DS18B20_init

     void DS18B20_measure(void) {//-----------------------------
------------------------

        m_err = 0;
        ds.reset();
        ds.select(addr);
        ds.write(0x44, 1);     // start conversion, with parasite
power on at the end
        delay(1000);          // time need for conversion
        present = ds.reset();
        ds.select(addr);
        ds.write(0xBE);       // Read Scratchpad

        //Serial.print("  Data = ");
        //Serial.print(present, HEX);
        //Serial.print(" ");
        for ( i = 0; i < 12; i++) {           // we need 12 bytes
resolution
          data[i] = ds.read();
          //Serial.print(data[i], HEX);
          //Serial.print(" ");
        }
        //Serial.print(" CRC=");
        //Serial.print(OneWire::crc8(data, 8), HEX);
        //Serial.println();

       if (OneWire::crc8(data, 8) != data[8]) {
            Serial.println("CRC is not valid!");
            m_err = 1;
        }

        // Convert the data to actual temperature
        int16_t raw = (data[1] << 8) | data[0];
        if (type_s) {
          raw = raw << 3; // 9 bit resolution default
          if (data[7] == 0x10) {
```

```
          // "count remain" gives full 12 bit resolution
          raw = (raw & 0xFFF0) + 12 - data[6];
        }
      } else {
        byte cfg = (data[4] & 0x60);
        // at lower res, the low bits are undefined, so let's
zero them
        if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution,
93.75 ms
        else if (cfg == 0x20) raw = raw & ~3; // 10 bit res,
187.5 ms
        else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375
ms
        //// default is 12 bit resolution, 750 ms conversion
time
      }
      celsius = (float)raw / 16.0;
      //Serial.print("  Temperature = ");
      //Serial.print(celsius);
      //Serial.print(" Celsius, ");
      //Serial.println();
    }//DS18B20_measure

    void Measure_SMT ()
       {
        float t_current, tj;
        int m_cycle, j, m;
        unsigned long dtime;
        Time_Heat_Dissipation = 0.0;
        j = 0;
        m = 0;
        tj = 0.0;

        Serial.println("Temperature of Soil measurement ...");
        for (m_cycle = 0; m_cycle<10; m_cycle++)
            {
              DS18B20_measure();
              if (m_err == 0)
                 {
                  tj = tj + celsius;
                  j++;
                  Serial.println(celsius);
                 }//if
              else {m++;}
             }//for

        if (j > 0 && m < 7)  { Soil_temperature = tj/j; }
        else {Soil_temperature = -21.0;}
        if (Soil_temperature > 0.0) {
        Serial.println("Heating ...");
        digitalWrite(pin_Heater, HIGH);
        delay(Heating_Time);                          // Time of
heating, ms
```

```
        digitalWrite(pin_Heater, LOW);
        Serial.println("Heat dissipation ...");
        dtime = millis(); // start time of Heat dissipation
        t_current = (Soil_temperature + 5.0);
        DS18B20_measure();
        if (m_err == 0) {t_current = celsius;}
        m_cycle = 0;

        while (t_current > (Soil_temperature + 1.0) && m_cycle <
250)
            {
             DS18B20_measure();
             if (m_err == 0) {t_current = celsius;}
             Serial.println(t_current );
             m_cycle++;
            }//while

        Time_Heat_Dissipation = (millis() - dtime)/1000.0;
        Serial.print("Time of Heat Dissipation = ");
        Serial.print(Time_Heat_Dissipation);
        Serial.println(", seconds");
       }//Soil_temperature > 0
    }// Measure_SMT
```

Програма для ПКЗ

```
    /*
      LoRa Half Duplex communication
      Control Relay Device
      MC: Wemos ESP8266

    */
    #include <SPI.h>              // include libraries
    #include <LoRa.h>
    #include <EEPROM.h>
    #define DARK  HIGH
    #define LIGHT LOW

    const long frequency = 433E6;  // LoRa Frequency
    const int csPin = 15;          // LoRa radio chip select
GPIO15(D8)->SS OF Lora module
    const int resetPin = 0;            // LoRa radio reset
GPIO0(D3)->RESET OF Lora module
    const int irqPin = 5;          // change for your board;
must be a hardware interrupt pin
    String outgoing;               // outgoing message
    String message = "";
    String Parameter_value_string;

    byte SystemID  = 0xA1;         // ID irrigation system
    byte SensorID  = 0xC1;         // ID this device
    byte MessageID = 0xDD;         // data, 0xAA setup, 0xCC
control
```

```
     byte SensorIN  = 0xB1;          // sensor Input
     byte ControlID = 0x11;          // ID control device
     byte TimerID   = 0x07;          // ID timer device
     float Parameter_num = 2.0;      // Soil moisture
     float Parameter_value;          // Parameter for control
     float Level_LOW   = 50.0;       // Low level of the Parameter
     float  Level_HIGH   =  90.0;          //  High  level  of  the
Parameter
     int Control_status = 0;         // Control position status 1-
ON/0-OFF.
     float Control = 1.0;            // Control: 1 auto/0 by hand

     // EEPROM coefficints
     int smem = 512;
     float kx = 0.0;
     int l = 6;
     char floatbufVar[7];
     int Setup_data;

     // Time
     String String_time = "00:00";
     String StartTime = "00:00";
     String StopTime  = "00:00";
     int Timer_status = 0;            // Timer position status 1-
ON/0-OFF.
     int i, j, k;
     long zi, zp;
     int pin_Led = 2;
     int pin_REL = 16;         // D0 Relay signal
     void ReadFromMem(void);
     void WriteToMem(void);

     void setup() {
       pinMode(pin_Led,OUTPUT);
       pinMode(pin_REL, OUTPUT);
       digitalWrite(pin_Led,DARK);
       digitalWrite(pin_REL, LOW);

       Serial.begin(115200);         // initialize serial
       while (!Serial);
       Serial.println(" ");

       // Reading setup coefficient from EEPROM
       //WriteToMem();    // need for first time
       ReadFromMem();

       // check if Timer control
       if   (   StartTime   ==   StopTime){Timer_status   =   1;
Serial.println("Timer is OFF");}
       else{Timer_status = 0; Serial.println("Timer is ON");}

       // override the default CS, reset, and IRQ pins (optional)
```

```
      LoRa.setPins(csPin, resetPin, irqPin);// set CS, reset,
IRQ pin

      if (!LoRa.begin(frequency)) {              // initialize
ratio at frequency MHz
        Serial.println("LoRa    init    failed.    Check    your
connections.");
        while (true);                          // if failed, do
nothing
      }
      Serial.println("LoRa init succeeded.");
    }

    void loop() {
      // parse for a packet, and call onReceive with the result:
      onReceive(LoRa.parsePacket());
    }//loop

    void sendMessage(String outgoing) {
      LoRa.beginPacket();                    // start packet
      LoRa.write(SystemID);                   // add destination
address
      LoRa.write(SensorID);                       // add sender
address
      LoRa.write(MessageID);                 // add message ID
      LoRa.write(outgoing.length());             // add payload
length
      LoRa.print(outgoing);                  // add payload
      LoRa.endPacket();                      // finish packet and
send it

    }//sendMessage

    void onReceive(int packetSize) {
      if (packetSize == 0) return;            // if there's no
packet, return
      // read packet header bytes:
      byte systemADR = LoRa.read();          // systemADR address
      byte sender = LoRa.read();             // sender address
      byte incomingMsgId = LoRa.read();      // incoming msg ID
      byte incomingLength = LoRa.read();      // incoming msg
length
      String incoming = "";
      while (LoRa.available()) {
        incoming += (char)LoRa.read();
      }

      if (incomingLength != incoming.length()) {     // check
length for error
        Serial.println("error: message length does not match
length");
        return;                                  // skip rest of
function
```

```
      }


      // if the recipient isn't this device or broadcast,
      if (SystemID != systemADR) {
        Serial.println("This message is not for me.");
        return;                                  // skip rest of
function
      }
      // if message from Timer
      if (sender == TimerID && incomingMsgId == SensorID) {
        String_time  =  incoming.substring(0,5);  //  this  is
current Time
        if(String_time >= StartTime && String_time < StopTime) {
           Timer_status = 1;
          }
        else {Timer_status = 0;}

      }//if Time
      // if message from Sensor and Control is Auto
      if (sender == SensorIN && Control == 1.0) {
      Serial.println("System: 0x" + String(systemADR, HEX));
      Serial.println("Sender: 0x" + String(sender, HEX));
      Serial.println("Message ID: " + String(incomingMsgId));
      Serial.println("Message        length:        "        +
String(incomingLength));
      Serial.println("Message: " + incoming);
      Serial.println("RSSI: " + String(LoRa.packetRssi()));
      Serial.println("Snr: " + String(LoRa.packetSnr()));
      Serial.println();

      // looking for Parameter_value by Parameter_num
      i = 0;
      j = 0;
      k = 0;

      i = incoming.length();
      k = incoming.indexOf("#",j);
      if (k >= 0) { incoming = incoming.substring(1,i); }

      i = incoming.length();
      k = incoming.indexOf("#",1);
      if   (k   >   0)   {   i   =   i   -   1;   incoming   =
incoming.substring(0,i); }

      for (i=0; i<Parameter_num; i++)
          {
            k = incoming.indexOf(",",j);
            Parameter_value_string = incoming.substring(j,k);
            j = k + 1;
          }//
      Parameter_value = Parameter_value_string.toFloat();
      Serial.println(Parameter_value);
```

```
        // Relay control
        // *** Check all combinations *****
        if (Parameter_value < Level_LOW && Timer_status == 1) {
            Control_status = 1;
            digitalWrite(pin_REL, HIGH);
            Serial.println("Relay ON");
           }

        if (Parameter_value > Level_HIGH || Timer_status == 0) {
            Control_status = 0;
            digitalWrite(pin_REL, LOW);
            Serial.println("Relay OFF");
           }
        // Sending message
        message = "#";
        message+=Control;
        message+=",";
        message+=Level_LOW;
        message+=",";
        message+=Level_HIGH;
        message+=",";
        message+=Parameter_value;
        message+=",";
        message+=Control_status;
        message+="#";
        delay(random(1000,3000));
        sendMessage(message);
        Serial.println("Sending " + message);
        }//if Sensor

      if (sender == ControlID && incomingMsgId == SensorID) {
        Serial.println("Control command");
        Serial.println("System: 0x" + String(systemADR, HEX));
        Serial.println("Sender: 0x" + String(sender, HEX));
        Serial.println("Message ID: " + String(incomingMsgId));
        Serial.println("Message         length:         "         +
String(incomingLength));
        Serial.println("Message: " + incoming);
        Serial.println("RSSI: " + String(LoRa.packetRssi()));
        Serial.println("Snr: " + String(LoRa.packetSnr()));
        Serial.println();
        int cs = incoming.indexOf(",");
        String control_string = incoming.substring(0,cs);
        zi = control_string.toInt();
        Serial.println(zi);
        int fs = cs + 1;
        cs = incoming.indexOf(",", fs);
        control_string = incoming.substring(fs,cs);
        Serial.println(control_string);

        if(zi    ==    1){    zp    =    control_string.toInt();
Serial.println(zp); SystemID = (byte)zp;}
```

```
       if(zi    ==    2){    zp    =    control_string.toInt();
Serial.println(zp); SensorID = (byte)zp;}
       if(zi    ==    3){    zp    =    control_string.toInt();
Serial.println(zp); SensorIN = (byte)zp;}
       if(zi    ==    4){    kx    =    control_string.toFloat();
Serial.println(kx); Control = kx;}
       if(zi    ==    5){    kx    =    control_string.toFloat();
Serial.println(kx); Parameter_num = kx;}
       if(zi    ==    6){    kx    =    control_string.toFloat();
Serial.println(kx); Level_LOW = kx;}
       if(zi    ==    7){    kx    =    control_string.toFloat();
Serial.println(kx); Level_HIGH = kx;}
       if(zi == 8){ Serial.println(control_string); StartTime =
control_string;}
       if(zi == 9){ Serial.println(control_string); StopTime =
control_string;}

       if(zi == 10){
          zp = control_string.toInt();
          Serial.println(zp);
          Control_status = zp;
          // Relay control
          if (Control_status == 1) {digitalWrite(pin_REL, HIGH);
Serial.println("Relay ON");}
          if (Control_status == 0) {digitalWrite(pin_REL, LOW);
Serial.println("Relay OFF");}
          }//7

       if(zi > 0 && zi <10){ WriteToMem(); }//if zi

           // Sending message
           message = "";
           message+=Control;
           message+=",";
           message+=Level_LOW;
           message+=",";
           message+=Level_HIGH;
           message+=",";
           message+=Parameter_value;
           message+=",";
           message+=Control_status;
           sendMessage(message);
           Serial.println("Sending " + message);
     }//if Control
     }//onReceive

     void ReadFromMem(void){
          Serial.println("Reading coefficients from EPROM ... ");
          EEPROM.begin(smem);
          j = 1*0;
          for    (i=0;    i<1;    i++)    {floatbufVar[i]    =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
```

```
          SystemID = (byte)kx;
          Serial.println(SystemID);

          j = l*1;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          SensorID = (byte)kx;
          Serial.println(SensorID);

          j = l*2;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          SensorIN = (byte)kx;
          Serial.println(SensorIN);

          j = l*3;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          Control = kx;
          Serial.println(Control);

          j = l*4;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          Parameter_num = kx;
          Serial.println(Parameter_num);

          j = l*5;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          Level_LOW = kx;
          Serial.println(Level_LOW);

          j = l*6;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          kx = atof(floatbufVar);
          Level_HIGH = kx;
          Serial.println(Level_HIGH);

          j = l*7;
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          StartTime = String(floatbufVar);
          Serial.println(StartTime);

          j = l*8;
```

```
          for     (i=0;     i<l;     i++)     {floatbufVar[i]     =
EEPROM.read(i+j);}
          StopTime = String(floatbufVar);
          Serial.println(StopTime);
          EEPROM.end();
          Serial.println("Completed");
     }// ReadFromMem()

     void WriteToMem(void){
       Serial.println("Writing data to EEPROM ... ");

       EEPROM.begin(smem);

           j = l*0;
           kx = (float)SystemID;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*1;
           kx = (float)SensorID;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*2;
           kx = (float)SensorIN;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*3;
           kx = Control;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*4;
           kx = Parameter_num;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*5;
           kx = Level_LOW;
           dtostrf(kx, l, 1, floatbufVar);
           for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

           j = l*6;
           kx = Level_HIGH;
           dtostrf(kx, l, 1, floatbufVar);
```

```
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          j = l*7;
          StartTime.toCharArray(floatbufVar,6);
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }

          j = l*8;
          StopTime.toCharArray(floatbufVar,6);
          for     (i=0;     i<l;     i++)     {EEPROM.write(i+j,
floatbufVar[i]); }
          EEPROM.commit();
          EEPROM.end();
          Serial.println("Completed");
          Setup_data = 0;
     }//WriteToMem
```

Програма для шлюзу

```
     /*
       LoRa Duplex communication
       Gateway Device
       MC: Wemos ESP8266
     */
     #include <SPI.h>               // include libraries
     #include <LoRa.h>
     const long frequency = 433E6;  // LoRa Frequency
     const int csPin = 15;             // LoRa radio chip select
GPIO15(D8)->SS OF Lora module
     const  int  resetPin  =  0;             //  LoRa  radio  reset
GPIO0(D3)->RESET OF Lora module
     const int irqPin = 5;             // change for your board;
must be a hardware interrupt pin
     String outgoing;               // outgoing message
     byte systemID =  0xA1;         // ID irrigation system
     byte deviceID =  0x11;          // ID this device 17 - this
is Gateway
     byte messageID = 255;          // ID destination

     void setup() {
       Serial.begin(115200);        // initialize serial
       while (!Serial);

       //Serial.println("LoRa Duplex");
       // override the default CS, reset, and IRQ pins (optional)
       LoRa.setPins(csPin,  resetPin,  irqPin);//  set  CS,  reset,
IRQ pin
       if (!LoRa.begin(frequency)) {               // initialize
ratio at frequency MHz
          Serial.println(String(deviceID)  +  ","  +  0);  //  if
failed
```

```
          while (true);                          // if failed, do
nothing
      }
      Serial.println(String(deviceID) + "," + 1); // if LoRa
init succeeded
    }

    void loop() {
      // reading the Command via Serial port form PC
      String serial_incoming = "";
      while (Serial.available()) {
        serial_incoming += (char)Serial.read();
        delay(10);
      }

      int ls = serial_incoming.length();
      if(ls > 0) {

        Serial.println(serial_incoming);
        int cs = serial_incoming.indexOf(",");
        String control_string = serial_incoming.substring(0,cs);
        int zi = control_string.toInt();
        messageID = (byte)zi;
        serial_incoming = serial_incoming.substring((cs+1),ls);
        String message = "";
        message+=serial_incoming;
        sendMessage(message);
        ls = 0;
        }//if serial_incoming
      // parse for a packet, and call onReceive with the result:
      onReceive(LoRa.parsePacket());
    }

    void sendMessage(String outgoing) {
      LoRa.beginPacket();                    // start packet
      LoRa.write(systemID);                  // system address
      LoRa.write(deviceID);                      // this device
address
      LoRa.write(messageID);                 // destination ID
      LoRa.write(outgoing.length());             // add payload
length
      LoRa.print(outgoing);                  // add payload
      LoRa.endPacket();                      // finish packet and
send it
    }

    void onReceive(int packetSize) {
      if (packetSize == 0) return;            // if there's no
packet, return
      // read packet header bytes:
      byte systemADR = LoRa.read();          // systemADR address
      byte sender = LoRa.read();             // sender address
      byte incomingMsgId = LoRa.read();      // incoming msg ID
```

```
      byte incomingLength = LoRa.read();      // incoming msg
length
      String incoming = "";

      while (LoRa.available()) {
        incoming += (char)LoRa.read();
      }
      if (incomingLength != incoming.length()) {     // check
length for error
        //Serial.println("error: message length does not match
length");
        return;                                  // skip rest of
function
      }
      // if the recipient isn't this device or broadcast,
      if (systemID != systemADR) {
        //Serial.println("This message is not for me.");
        return;                                  // skip rest of
function
      }
      // if message is for this device, print details:
      //Serial.println("System: 0x" + String(systemADR, HEX));
      //Serial.println("Sender: 0x" + String(sender, HEX));
      //Serial.println("Message ID: " + String(incomingMsgId));
      //Serial.println("Message       length:       "       +
String(incomingLength));

      Serial.println(String(sender) + "," + incoming + "," +
String(LoRa.packetRssi())+ "," + String(LoRa.packetSnr()));
      //Serial.println("RSSI: " + String(LoRa.packetRssi()));
      //Serial.println("Snr: " + String(LoRa.packetSnr()));
      //Serial.println();
    }//receive
```