Зміст

Завдання

Розробити програму передачі даних з мікроконтролера MSP430 на персональний комп’ютер за допомогою USB(UART) ,а також обробити їх, використавши мову програмування Java з використанням бібліотеки jSSC для роботи з ком портом, реалізувати деяке графічне відображення. Створити свій протокол передачі даних який забезпечить максимальну гнучкість при передачі різних параметрів мікроклімату з одного контролера.

Вступ

У зв’язку з виникненням продуктової проблеми в світовому масштабі ,а також зменшенням посівної площі земель , людство починає розширювати екстенсивні способи вирощування рослин.

В останній час стрімко розвиваються системи управління мікрокліматом та якісна підтримка параметрів мікроклімату, що дозволяє значно збільшити врожайність, при мінімальних затратах. Одним із головних параметрів є ефективне використання енерго-ресурсів, що дозволяє суттєво зменшити собівартість врожаю . Але на даний час енергоносії є доволі дорогими і тому доцільно використовувати альтернативні(природні) джерела енергії, а саме: енергія сонця,вітру ,термальна та використання біо-топлива. Це дозволяє зменшити затрати на енергоресурси. Завдання системи управління полягає у ефективному використанні природних ресурсів.

Актуальність даної теми полягає в створенні систем з раціональнішим використанням альтернативних та відновлювальних енергоресурсів. Існує велика кількість альтернативних видів енергії які використовуються не у повному обсязі тому автоматизації технологічних процесів в тепличному господарстві дозволить створити незалежну систему мікроклімату, яка вирощуватиме різні рослинні культури в будь-який період року.

Об’єкт

Об’єктом дослідження є регульоване середовище тепличного господарства

Предмет дослідження

Предметом дослідження є дослідження можливості забезпечення зміни регулюючих параметрів.

ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Автоматична незалежна система вимірювальних параметрів контролю мікроклімат теплиці в заданих межах кліматичних норм.

ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ

Розробка системи управління мікрокліматом для підтримання заданих параметрів мікросередовища в різних видах теплиць базується на максимальному використанні альтернативних видів енергії для забезпечення заданих параметрів мікроклімату. Для підтримки незалежності системи управління ,нажаль, не можливо використати повністю не залежне джерело енергії але його можна звести до мінімуму. Серед параметрів які впливають на якість теплиці можемо виділити наступні.

Параметри, що впливають на якість теплиці.

Кліматичні системи забезпечують комплексний контроль за мікрокліматом теплиці, а саме управління температурою, вологістю, вентиляцією, обігрівом та CO2, поливом, рециркуляцією повітря та іншими системами. Управління цими параметрами здійснюється в автоматичному та напівавтоматичному режимі.

Для цього в різних зонах теплиці монтується контрольно-вимірювальне обладнання: датчики температури, вологості, освітлення, тиску, тощо.

Окремо назовні теплиці встановлюється метеовежа. Для контролю та управління параметрами мікроклімату використовуватися кліматична система з встановленим програмним забезпеченням. В залежності від технології вирощування та кількості культур, які одночасно вирощуються в теплиці, можна обрати різні моделі системи клімат-контролю, що забезпечують управління мікрокліматом в одному або одночасно в 4-х окремих блоках тепличного комплексу.

Обігрів та CO2

До вибору системи опалення теплиць завжди слід підходити з урахуванням кліматичних умов регіону, наявності та вартості енергоресурсів (палива), планів щодо експлуатації протягом року, а також культур, які будуть вирощуватися. Основне завдання системи опалення - це забезпечення заданого температурного режиму особливо в осінньо-зимовий період. Також за допомогою обігріву можна контролювати такий параметр як вологість в теплиці та рівень **CO2**.

Вуглекислий газ дає ефект коли рослині достатньо і світла і поживних речовин. Оптимальним вважається рівень CO2 в діапазоні від 1000 ppm до 1600 ppm. Якщо теплиця не

обладнана системою подачі CO2, вона потребує значного провітрюванні, щоб вуглекислий газ надходив з навколишнього середовища. У більшості промислових теплиць, вуглекислий газ отримують шляхом спалювання побутового газу. Для невеликих установок можна отримувати вуглекислий газ і іншими способами[2].

- Балон зі стисненим вуглекислим газом, редуктором і соленоїдом.
- Система, заснована на бродінні:

  На 1.5 літрову пляшку води береться 100г цукру і пів чайної ложки дріжджів. можна додати соду на кінчику ножа. Друга пляшка в системі (з чистою водою) служить для уловлювання ефірних масел і як страховка якщо "брага" почне пінитися і витікати.
- 3. Система, основана на хімічній реакції.

При правильному використанні CO2 можна збільшити врожайність до 40%.

Повітря теплиці, як і атмосферний, складається з газів: азот, кисень, вуглекислий газ, пари води, домішки інших газів. І хоча вуглекислий газ можна віднести мінеральному живленню рослин, при цьому необхідно розглядати як макроелемент, поряд з азотом, фосфором, калієм і кальцієм, так як рослин на 95% складаються з вуглецю. Основним джерелом вуглецю для рослин є вуглекислий газ повітря. Тому ми будемо розглядати концентрацію вуглекислого газу як параметр мікроклімату.

Споживання вуглекислого газу - це ріст рослини. Інтенсивність і продуктивність фотосинтезу зростає на 50% при підвищенні концентрації вуглекислого газу в повітрі теплиці з 300 до 900 ppm.

На практиці концентрація вуглекислого газу більше 700 ppm потрібна тільки за умови отриманні достатньої кількості світлової енергії.

На мою думку, найбільш досконалою є система підживлення рослин рідким вуглекислим газом. Вона забезпечує необхідну чистоту газу і як правило вільна від шкідливих домішок (CO, NO2, SO 2). Установка рідкої вуглекислоти дає можливість регулювання не тільки об'ємом подачі газової суміші, як відбір газу від котла або електрогенератора, а й концентрацією подається в теплиці газової суміші.

Чи не мале значення має утилізація тепла виробленого при спалюванні палива з метою отримання вуглекислоти для підживлення рослин. Актуально це в літній період, особливо в південних регіонах країни.

З точки зору економії ресурсів можна рекомендувати проводити підживлення CO2 в зимовий (холодне) час використовуючи відходять гази котлів та електрогенераторів, а в літній (тепле) час - установки з рідким CO2.

Однак необхідно пам'ятати про забруднюючих речовинах, що впливають на якість кольорів. В основному це чадний газ (CO), який шкідливий для людей і квітів.

Гранично-допустима концентрація становить 20мг/м3 повітря теплиці. З цього показника налаштовані всі пристрої (котли та електрогенератори) застосовувані в теплицях, які спалюють вуглеводневе паливо з метою отримання CO2.

При освітленості 6000 люкс не потрібна концентрація вуглекислого газу в 1000 ppm, а буде достатній рівень 500-600 ppm. При цьому треба враховувати, що в такій ситуації верхні листки продукують продукти фотосинтезу, а нижні їх з'їдають[2].

Водяний обігрів

Водяний обігрів підходить для вирощування високих культур , як томати, огірки, перець, тощо. Тепличні водогрійні та парові котли характеризуються великим об'ємом води, що дозволяє справитися с різкими змінами зовнішніх температур, і, відповідно, с перепадами температури в теплиці.

В теплицях використовується система з розділенням контурівКількість контурів залежить від різноманіття культур, що вирощуються в теплиці. Зазвичай проектуються три контури обігріву. Два контури надґрунтового обігріву та один загальний контур підкрівельного обігріву. Такий метод дозволяє вирівняти температуру по всій площі теплиці.

В зимовий період система вимірює температуру від -30,-20С щоб забезпечувати оптимальні умови та контролювати стан температури в середині щоб уникнути різких перепадів температури та підтримувати задані границі температури.

Сонячні Колектори

Найбільш потужним джерелом енергії для людства є Сонце. Річна кількість сонячної енергії майже в 15 000 разів перевищує потреби населення нашої планети, проте лише незначна її частина використовується на господарські потреби. Для перетворення сонячної енергії в теплову використовують сонячні колектори (геліосистеми).

Сонячний колектор (геліоколектор) – це пристрій, який призначений для поглинання сонячної енергії, яка переноситься видимим та ближнім інфрачервоним випромінюванням та для подальшого її перетворення в теплову енергію, придатну для використання.

Насправді суть роботи сонячних колекторів доволі проста. Будь-який сонячний колектор, незалежно від його типу чи конструкції, перетворює енергію Сонця в теплову енергію для опалення, гарячого водопостачання, нагрівання басейну тощо. Втім, геліосистеми з використанням високоефективних вакуумних сонячних трубок здатні працювати цілий рік на відміну від плоских геліоколекторів. Вакуумна теплова трубка виготовляється зі спеціального зміцненого боросилікатного скла. Зовнішня труба такого колектора є прозорою, а внутрішня - покрита високоякісним селективним покриттям, яке забезпечує максимальне поглинання сонячного тепла при мінімальному рівні рефлекції (тобто, мінімальному рівні відбиття сонячних променів назад у атмосферу). Для уникнення теплових втрат між зовнішньою та внутрішньою трубками знахаходиться вакуум. Для того, що підтримувати вакуум, застососвують барієвий газопоглинач, який в виробничих умовах підлягає впливу високих температур. Через це нижній край вакуумного термосу покривається шаром чистого барію, який поглинає CO, CO_2, N_2, O_2, H_2O та H_2, що можуть виділятися з труби в процесі зберігання та експлуатації. Цей шар є дуже добрим візуальним детектором стану вакууму в трубі геліоколектора.

Цебто, коли вакуум порушується, барієвий шар зі сріблястого робиться білим. Такий індикаторний механізм дає можливість легко визначити, чи ціла труба в вакуумному сонячному колекторі, а чи має тріщину.

Абсорбування сонячного тепла проходить у мідній трубці, яка розташована всередині вакуумної труби. Спосіб передачі тепла від мідної трубки до головного теплопроводу сонячного колектора також простий. Мідна труба є порожнистою і містить всередині запатентовану неорганічну й зовсім нетоксичну рідину. При нагріванні ця рідина закипає і починає випаровуватися. Це відбувається навіть при мінусових температурах, окільки в трубці, як ви пам"ятаєте, створено вакуум. Нагріта пара піднімається до верхнього наконечника (конденсатора) теплової трубки, де передає тепло теплоносію (антифризу), що циркулює в трубі теплопровода. Потім пара конденсується й стікає вниз - процес починається знову. Сонячний водонагрівач з вакуумними трубами показує задовільні результати навіть у хмарні дні, тому що труби сонячного колектора здатні поглинати енергію інфрачервоних променів, які проходять через хмари.

Завдяки ізоляційним властивостям вакууму вплив вітру та низьких температур на роботу вакуумних трубчатих геліоколекторів абсолютно нівелюється у порівнянні з плоскими

геліоколекторам .Системи на основі вакуумних сонячних колекторів успішно нагрівають воду, навіть коли на вулиці -35°С.

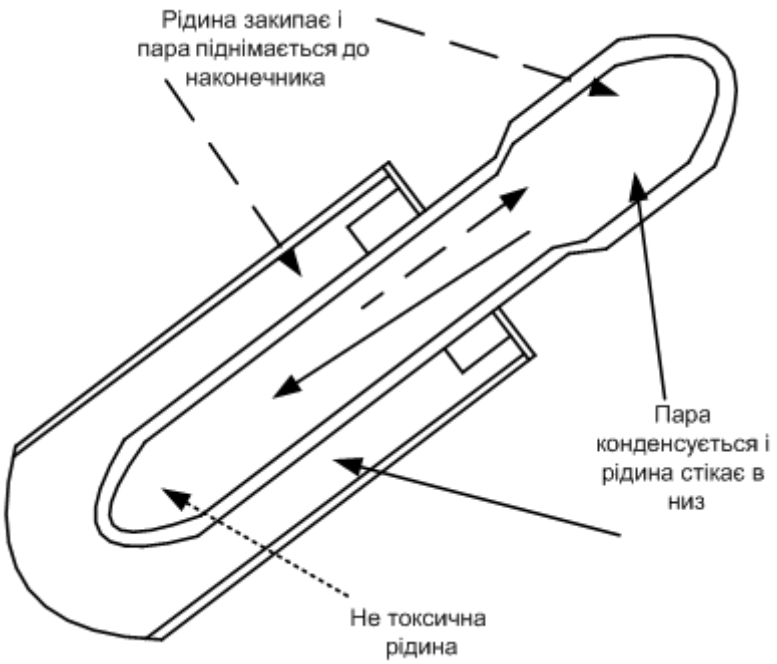


Рис.4.2 Теплообмін в тепловій трубці сонячного колектора

Труби геліоколектора мають круглу форму, завдяки чому кількість сонячної енергії, яка падає на сонячний колектор практично не змінюється протягом дня(**Рис.4.2**). Саме тому загальна кількість сонячного випромінювання, яке поглинає геліоколектор є значно більшою, якщо порівнювати таку систему з пласким сонячним колектором. Така форма труб забезпечує чудове поглинання енергії оскільки сонячні промені завжди падають на поверхню вакуумного сонячного колектора строго під прямим кутом, при цьому відбивання зводиться до мінімуму. Труби розміщуються в колекторі паралельно одна одній, кут їх нахилу відносно горизонту залежить від географічної широти місцевості, де встановлюється сонячна система опалення.[3]

Правильно орієнтовані трубки протягом дня пасивно рухаються за сонцем.

Такий сонячний водонагрівач зовсім не вимагає обслуговування під час експлуатації(**Рис4.3**).

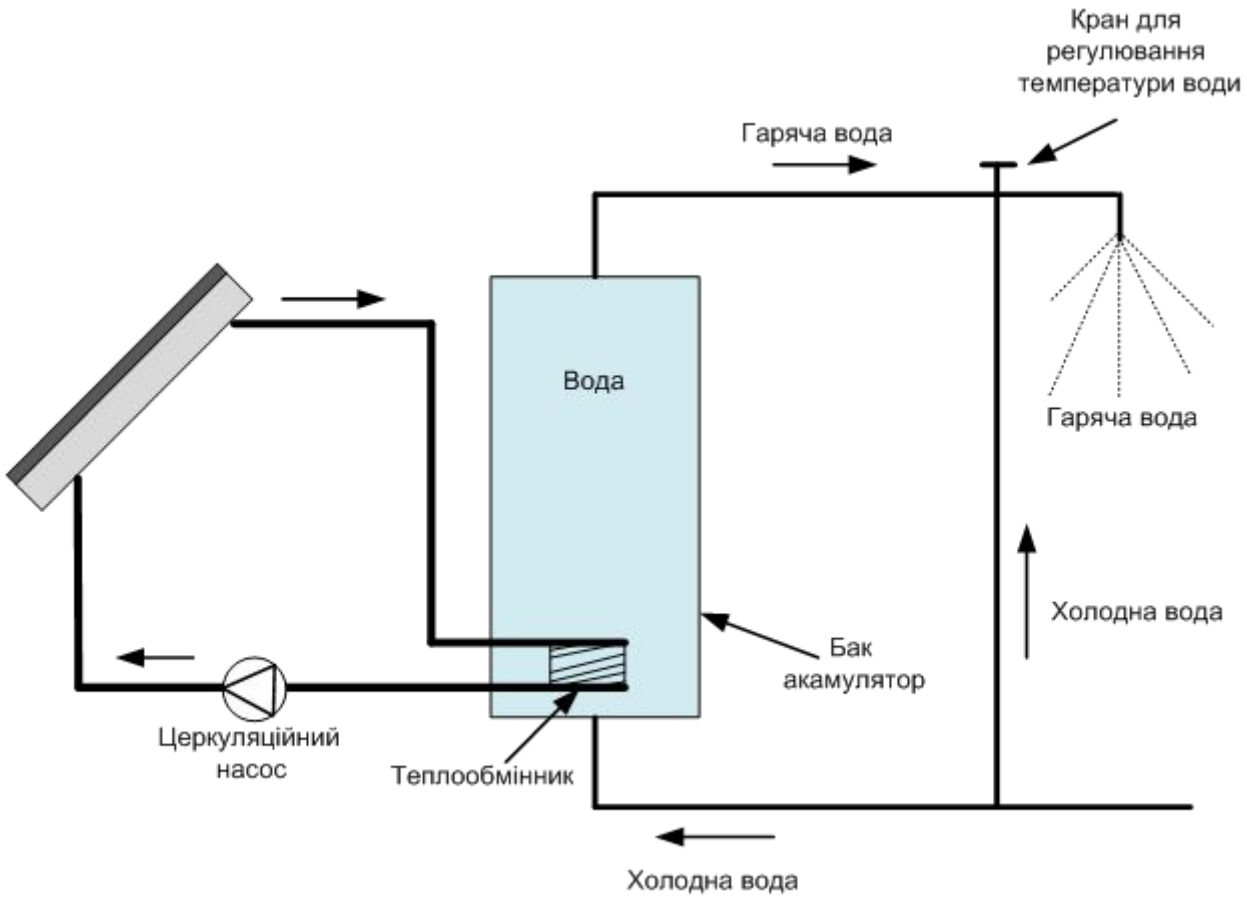


Рис4.3 Система управління обігріву та подачі води з розділенням контурів

Сонячна система також є простою у ремонті: якщо виникне така потреба, вакуумну трубку можна легко поміняти, не зупиняючи при цьому сонячний колектор. При необхідності трубки можна додавати (при недостачі тепла) чи частково знімати (якщо тепло поглинається в надлишку), зменшуючи геліополе, що, зауважте, абсолютно неможливо в геліосистемах з пласкими колекторами[3].

Геліоколектори відмінно справляються з завданням забезпечення тепличного господарства гарячою водою, підігрівом води в резервуарах, працюють в системах вентиляції та опалення. За для ефективного використання ресурсів система вимірює,контролю температуру води від 1-150С та її рівень в резервуарі, для забезпечення контролю використовується діапазон вимірювання температури води від -30 до150 С

,а також ведеться контроль витрат води та формування погодинного архіву значень обсягу й витрат. На показники витрат води впливає швидкість подачі води та діаметр труб.

Повітряний обігрів

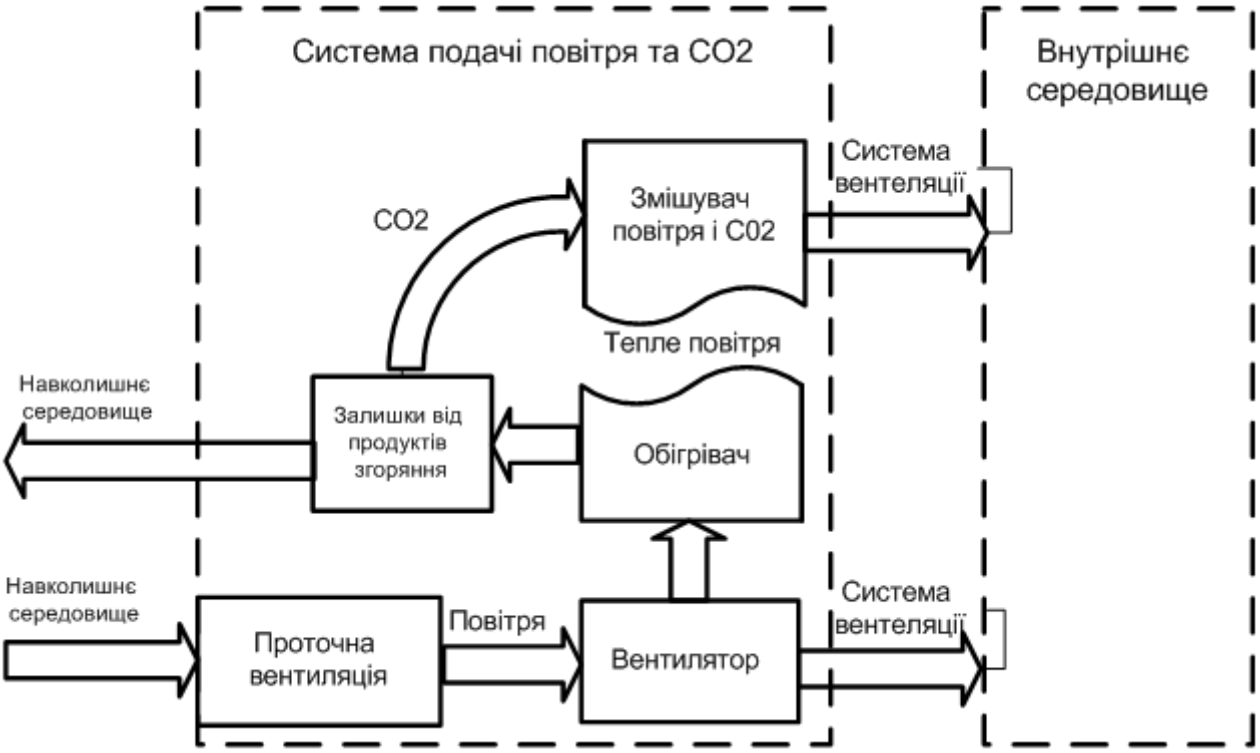


Рис 4.4 Система обігріву повітря

Використання у теплицях системи повітряного обігріву є більш економним варіантом, але має низку специфічних особливостей. Зазвичай ця система використовується в поєднанні з водяним обігрівом в регіонах, де мінімальні температури досягають -20 °C и нижче.

Однак в місцевостях з м'яким кліматом повітряний обігрів теплиць може використовуватися як основний. Система повітряного обігріву реалізується на базі повітронагрівача, що працює на газі, рідкому чи твердому паливі, повітроводів та системи вентиляторів. При вирощуванні високих рослин система повітроводів забезпечує подачу теплого повітря спрямовану в ряди з рослинами.

При вирощуванні низьких культур циркуляція теплого повітря від теплогенераторів забезпечується системою рециркуляційних вентиляторів. Система виде контроль температури повітря в приміщені, теплиці ,а також вимірює рівень CO2 і при необхідності збільшує його. Також ведеться контроль вологості повітря від 0-100%, для його подальшого регулювання**.**

Сонячний обігрів

Один з найбільш перспективних способів обігріву теплиць і парників на дачній ділянці - використання геліоенергії. Загальні принципи парникового ефекту, що лежить в основі сонячного опалення теплиці, добре відомі. **Обігрів парника** відбувається завдяки тому, що сонячні промені, що проходять через прозорепокриття, нагрівають грунт і предмети, від яких, у свою чергу, розігрівається повітря. Утримання тепла всередині теплиці забезпечується надійністю конструкції і якістю укривного матеріалу(**Рис 4.5**).

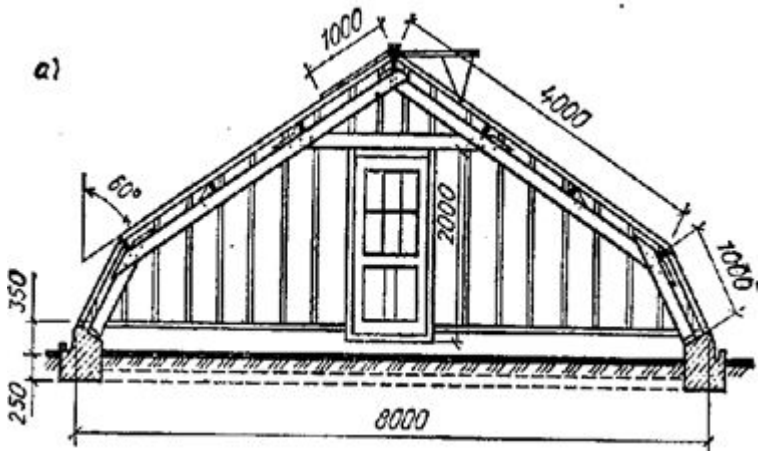


Рис 4.5 Конструкції теплиць для сонячного обігріву

Рівнем сонячного випромінювання керувати неможливо. Завдання дачника полягає в тому, щоб максимально ефективно використовувати те, що надає природа.

Ефективність використання сонячної енергії залежить відмісцерозташування теплиці та її орієнтації відносно сторін горизонту, форми теплиці і матеріалу покриття, застосування додаткових способів акумулювання енергії.

Як відомо, найменші тепловтрати - у кулястих об'єктів. Найбільш наближені до такої форми арочні теплиці з округлим склепінням і у вигляді шатра. Варто, однак, врахувати, що в теплицях з похилими стінками біля них неможливо вирощувати рослі рослини.

Практично весь потік сонячної енергії надходить в теплицю з південної сторони. Виходячи з цього, головна вимога до північної стінки - зменшення втрат. Для цього її можна зробити повністю непрозорою, утеплити і пофарбувати зсередини білою глянсовою фарбою або навіть покрити фольгою[4].

Світловий режим рослин в основному характеризується періодом освітлення (довжиною дня) і кількістю світла, яке виражається фізіологічною радіацією (350...750нм) або фотосинтетичною активною радіацією – ФАР (380...710нм). Так в південних районах світловий день в період вегетаційного сезону менший ніж середній полосі, але кількість сонячної радіації

на півдні більше (чим дальше від екватора, тим більше променистої енергії поглинається атмосферою. Реакцію рослин на період освітлення називають фотоперіодизмом. Тобто, з рухом з півдня на північ, краще ростуть і розвиваються ті рослини, філогенез яких протікав в умовах довгого дня. Південні екзоти, навпаки, краще розвиваються при більш коротшому дню. В результаті виникла фізіологічна класифікація – рослини короткого і довгого дня.[5]

Однак є і нейтральні види, які успішно ростуть як при короткому так і при довгому дню. До рослин короткого дня відносять жоржину, канни, настурцію, хризантеми, амарант,шальвію. Довгоденні рослини: айстри, фіалки, гладіолуси, гортензія, дельфініум та ін. Нейтральні види - тюльпан, лілія, нарцис, пеларгонія, цикламен, цинія та ін.

Як бачимо, вимоги до форми теплиці досить суперечливі. Орієнтуючись на «золоту середину», найбільшзручною та ефективною є теплиця у формі трапеції з високою утепленій північній стінкою, більш низькою південній і похилим дахом.

Електричний обігрів

Для обігріву тепличного грунту використовується спеціальний нагрівальний кабель, укладання якого може відбуватися двома способами: шляхом закладки нагрівальної системи безпосередньо в шари грунту або ж шляхом установки нагрівального кабелю у бетонну стяжку, на який зверху насипається родючий грунт. В обох випадках з метою зменшення тепло втрат бажано використовувати утеплювач.[6]

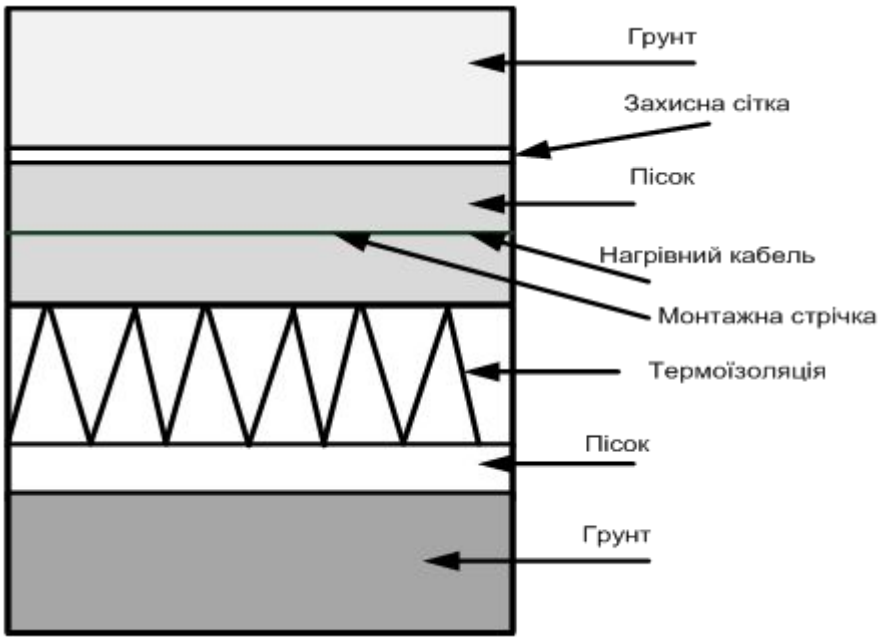


Рис 4.6 Структура електричної системи обігріву грунту

Закладка нагрівального кабелю в шари грунту передбачає використання спеціальної металевої сітки з дрібними отворами.

Сітка укладається зверху виложенного по всій поверхні кабелю. Це забезпечить збільшення експлуатаційного терміну всій нагрівальної системи. У разі закладки кабелю у бетонну стяжку, яка володіє більшою однорідністю і густиною, ніж грунт, коефіцієнт тепловіддачі істотно збільшується.

Крім того, таке утеплення грунту запобіжить можливість механічного або хімічного пошкодження нагрівального кабелю в ході обробки грунту. Відповідно система контролює температуру ґрунту як в кореневій системі так і на поверхності грунту та виде моніторинг використаної електроенергії. За для забезпечення оптимальних умов температура вимірюється в діапазоні від -20 до +50 С. А також ведеться конт роль вологості грунту від 0-100%.

Біологічний обігрів

Застосовується для обігрівання парників, теплиць і утепленого грунту. Органічні речовини, які швидко розігріваютьсяі виділяють велику кількість теплоти, називають біопаливом. До них належить гній, побутове сміття, волога і загнила солома, відходи деревообробної промисловості (тирса, кора), листя, слабкорозкладений торф, тощо. Найціннішим біопаливом є кінський гній. Температура його на 7-8-й день після закладання у парники досягає 60-75℃. А через 45-50 днів знижується до 30℃. Цей вид палива найбільш доцільно використовувати для закладання ранніх парників.

Заготівлю біопалива починають восени . Гній складають у бурти і добре ущільнюють , щоб запобігти передчасному розігріванню. На одну парникову раму заготовляють 0,5-0,9 тон гною залежно від строку використання . На 1га теплиці або утепленого грунту заготовляють 2-3 тис. тонн.

Розігрівати біопаливо починають за 8-12 днів до закладання парників. Його перебуртовують і складають нещільно, щоб забезпечити доступ повітря. У парники біопаливо закладають за 7-12 днів до сівби чи висаджування розсади і вкривають рамами, матами.

Для закладання наземних парників ділянку очищають від снігу і вкладають біопаливо шаром 40-60 см. На нього встановлюють короби, додають біопаливо, накривають рамами і матами. Правильно закладене в парники біопаливо розігрівається через 4-6 днів.

У теплицях і утепленому грунті біопаливо закладають у завчасно підготовлені заглиблення, або на поверхню грунту під грядки. Останім часом для біологічного обігріву теплиць почали використовувати солому.

Полив

Широкий вибір іригаційних систем включає в себе: обладнання для крапельного поливу, дощування, туманоутворення, поливу методом підтоплення. Незалежно від обраного типу поливу, для функціонування системи необхідно облаштувати так звану іригаційну кімнату для підготовки води, приготування, зберігання маточних розчинів, добрив та управляння процесами поливу та живлення рослин.

Крапельний полив

Рис 4.8 Система крапельного зрошення

Для дозованого поливу рослин використовується система крапельного зрошення(**Рис 4.8**), яка найкращим чином підходить для вирощування високих рослин таких як томати, огірки, перець, тощо.

Вона складається з пластикових труб, шлангів та крапельниць. Існує декілька видів таких систем, всі вони забезпечують оптимальний полив та подачу поживного розчину індивідуально кожній рослині. Ведеться контроль розміру капель 50-150 крон при тиску 3,0атм. та погодинні витрати.

Полив дощуванням

Для вирощування невисоких культур (розсади, зелені, декоративних рослин) використовується система дощування(**Рис 4.8**), в якій вода подається на зрошувану ділянку у вигляді дощу спеціальним дощувальним апаратом, який викидає струмінь води у повітря і розпилює її на краплі.

Рис 4.8 Система поливу дощуванням

Система дощування складається з постійного трубопроводу, апарату з пересувним трубопроводом та дощувального апарата, який пересувається по тросам під стелею теплиці.

При вирощування культур особливо вимогливих до підтримання високої вологості повітря в теплиці використовується система туманотворення. Вона складається зі спринклерів, встановлених над рослинами. Вода до спринклерів подається по пластиковим або сталевим трубам. Тиск, що забезпечується насосною групою, розпилює воду на дрібні краплини[7]. При такому підході одним із основних показників є контроль діаметру розпилення 2,0-4,0м при тиску 1,0-4,0 атмосфери та напрям розпорошення води (горизонтальний/вертикальний)

Полив підтопленням (заливні столи/підлога)

Варіантом поверхневого поливу в закритому ґрунті є система підтоплення. Вона може бути реалізована як у вигляді заливної підлоги, так і у вигляді заливних столів. Система працює за принципом тимчасового затоплення субстрату з кореневою масою та наступним відводом води. Операція затоплення/зливу воли виконується за допомогою насосів, які закачують поживний розчин з загального резервуару у піддон (на підлогу), де вирощується культура, потім залишки розчину повертаються назад в загальний резервуар самопливом. Необхідна циклічність циркуляції розчину забезпечується таймерами.

Система виде постійний контроль за рівнем води 0-5см. та за кількістю поживних речовин в ній. Така система дозволяє більш економічно використовувати воду та мінеральні добрива. В Україні система використовується для вирощування розсади та декоративних рослин.

Дренаж

Невикористана рослинами вода виводиться з теплиці за допомогою системи лотків, каналі та/або труб та спрямовується в дренажний колектор, потім за допомогою насосу

направляється за межі теплиці. За Вашим бажанням може бути встановлена система повторного використання води, після її очищення та дезінфекції. Параметри такої води ретельно аналізуються комп'ютером перед додаванням її в бак з водою для поливу.

Вентиляція

Для забезпечення необхідних параметрів температури та вологості в теплиці використовується система вентиляційних фрамуг та рециркуляційних вентиляторів. Природне провітрювання забезпечується як вентиляційними фрамугами розташованими на даху (покрівельна вентиляція) так и підйомними фрамугами на бокових та фронтонних стінках теплиці

Кут підняття фрамуг та площа вентиляційного отвору регулюється в залежності від температури повітря, швидкості вітру та опадів.

Управління відкриванням та закриванням фрамуг забезпечується автоматично системою кімат-контролю або за допомогою ручного приводу

Для **штучного переміщення повітря** в теплиці з метою вирівнювання температури, активації фізіологічних процесів в рослинах, ліквідації зон з підвищеною вологістю, особливо в періоди, коли природна вентиляція через фрамуги неможлива, застосовують рециркуляції вентилятори.

Система вентиляції допомагає контролювати рівень CO2 від 0-40% ,а також вологість повітря від 30-100% , також регулює температуру в теплиці ,як в літній так і зимній стан.

Розробка блоків вимірювання

Канал виміру CO2 який зображено на рисунку 7.5

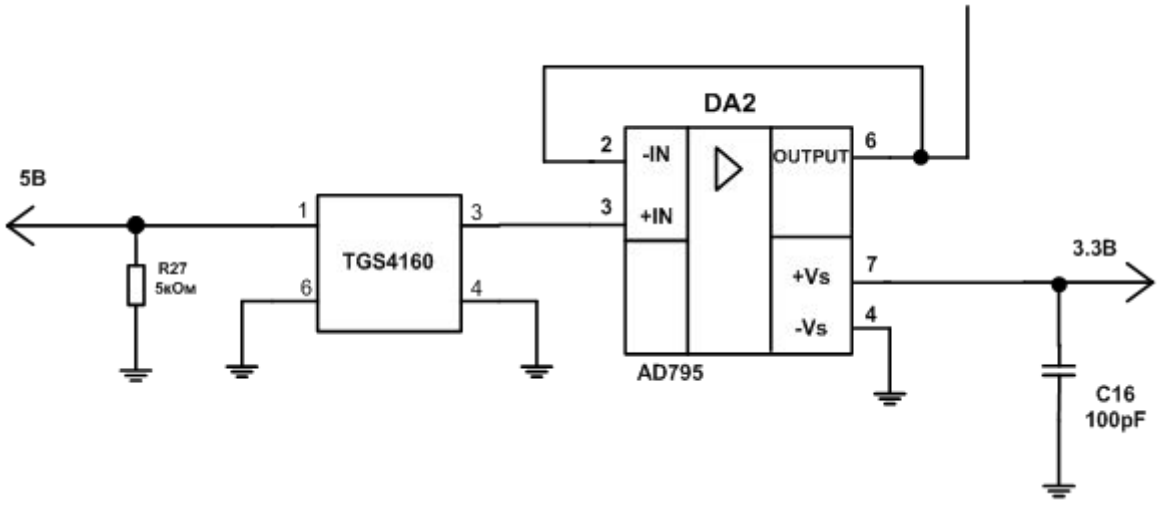


Рисунок 7.5 Канал виміру CO2

Вміст CO2 занходиться по формулі 7.4

EPC = Ec– (RT/2F)ln(PCO2) (7.4)

де Ec-константа, R - універсальна газово стала, F - постійна Фарадея, T - абсолютна температура (К), CO2-парціальний тиск вуглекислого газу.

Оскільки вимірювання газу основується на термопарі то для підтримання постійної температури робочого елемента підводиться живлення 5В, але щоб уникнути похибок від нагрівання потрібно подавати ток 0,3мкА.Тому з формули 7.5 ми отримаємо номінал резистора R27.

Виходячи з ряду E192 слідує, що R27 = 5,05кОм , допустиме відхилення ±5 %.

Вихідний сигнал датчика (EPC) перетворюється за допомогою операційного підсилювача з високим імпедансом (> 100 ГОм) і малим струмом зміщення (<1 пА) (AD795).

Розрахунок реального коефіцієнта підсилення за формулою 7.6. Так як вихідна напруга буде 30 мВ то потрібно підсилити сигнал в 100 раз.

$$K = \frac{K_{en}}{1 + K_{en} * \frac{1}{K}} = \frac{10000}{1 + 10000 * 0{,}01} = 99{,}99 \quad (7.6)$$

Блоки вимірювання освітленості та температури основуються на використанні цифрових датчиків STTS75 (температура) Рисунок 7.6

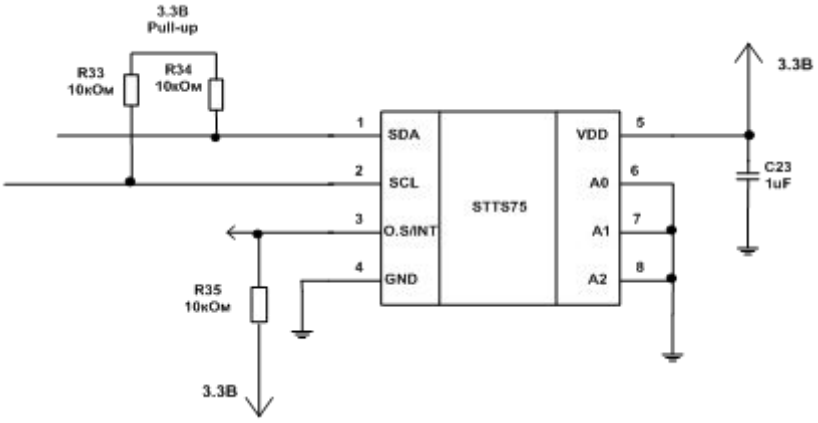


Рисунок 7.6 датчик температури STTS75

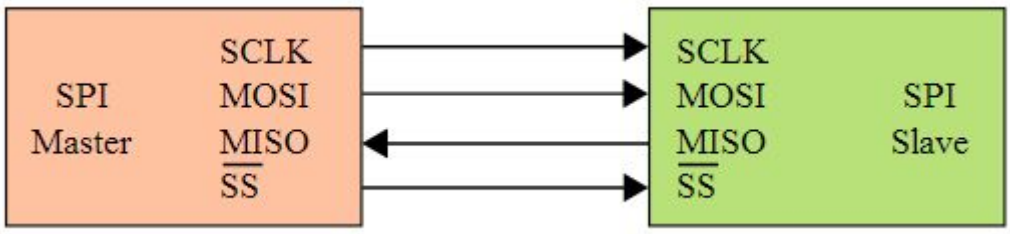
UART

UART (англ. *universal asynchronous receiver/transmitter* — універсальний асинхроний приймач/передавач) — тип асинхроного приймач-передавача, компонент комп'ютерів та периферії, що переводить дані між паралельною та послідовною формами. UART звичайно використовується спільно з іншими комунікаційними стандартами, такими як EIA RS-232.

UART це звичайно окрема мікросхема чи частина мікросхеми, що використовується для з'єднання через комп'ютерний чи периферійний послідовний порт. UART зараз загалом включені в мікроконтролери. Здвоєний UART (Dual UART або DUART) об'єднує два UART в одній мікросхемі. Багато сучасних мікросхем сьогодні випускаються з можливістю комунікації в синхронному режимі, такі прилади називають USART. Ряд стандартних швидкостей: 300; 600; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600 бод. Відстань 15м і більше..визначається багатьма параметрами.

4.2.2 SPI

(англ. *Serial Peripheral Interface, SPI bus* —послідовний периферійний інтерфейс, шина SPI) — послідовний синхронний стандарт передачі даних в режимі повного дуплексу, розроблений фірмою Motorola для забезпечення простого сполучення мікроконтролерів та периферії



Малюнок 4.3.1 SPI шина: один ведучий та один ведений

- Швидкість передачі даних до 52Мбіт / с;
- Максимальна частота 50 МГц;
- Кількість адресованих пристроїв необмежено;
- SPI-Serial Peripheral Bus (шина для підключення зовнішніх пристроїв);
- MOSI-послідовний ввід (дані від «помічника» до «майстру»);
- MISO - послідовний висновок (дані від «майстра» до «помічникові»);
- SCLK-Slave CLK (тактирование послідовного зв'язку (синхронізація));
- Повнодуплексний 3-дротовий синхронний обмін даними.
- Режим роботи провідний або ведений.
- Обмін даними з переданими першими старшим або молодшим бітами.
- Чотири програмовані швидкості обміну даними.

В даній системі використовується для взаємодії багатьох датчиків на одному мікроконтролері.

Моделювання

Для створення температурного каналу використовуються датчики KTY81/110-120

Дані приймаються допомогою вбудованого 10ти розрядного АЦП ,а потім передаються на персональний комп'ютер за допомогою USB проводка ,який йде в комплекті з MSP430 Launchpad. Подальша обробка даних здійснюється з використанням мови програмування Java оскільки вона є безкоштовною. (Також можна використати програмне середовище LabVIEW. Робота з портами здійснююся за допомогою модуля VISA який є вбудований в програму(LabVIEW) або можна додатково завантажити на сайті виробника. Також доволі зручними програмами для раннього тестування є Serial Oscilloscope, вона є вузько напрямлена але безкоштовна і зручна.)

Дана модель мікроконтролера(Рисунок 2) не має на своєму борту UART тому доводиться реалізовувати програмний UART.

Програмний UART контролера MSP430G2231 або MSP430G2553 та інших з серії MSP430G2XXX.

Для MSP430G2553 програмний UART не є необхідним адже він підтримує цей протокол на відміну від 2231 на борту якого SPI і I2C тому доводиться використовувати програмний UART на основі використання Timer A для зв'язку з персональним комп'ютером чи ноутбуком (тимбаче якщо вони не мають COM портів) з використанням USB кабеля який йде в комплекті з MSP430 LaunchPad .

Для чого це потрібно? По-перше , апаратний UART є далеко не на всіх контролерах.

Короткий опис

Так буде виглядіти посилаймий біт.

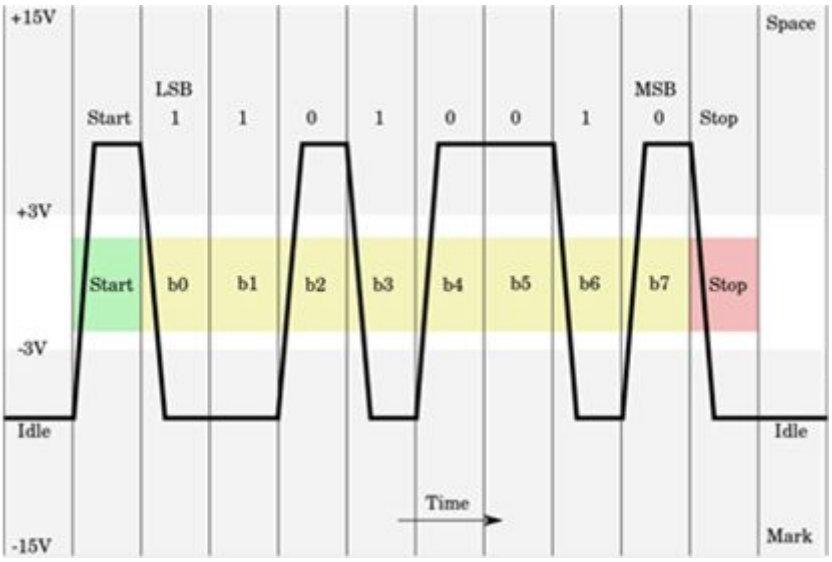


Рисунок 1.Восьмібітний формат даних, без перевірки на парність, з одним стоповим бітом.

Start - старт біт, B0 ... B7 - біти даних, Stop - стоп біт. У стандарті RS232 логічна одиниця називається "mark", при цьому рівень сигналу на лінії -5 ..-15В. Логічний нуль в стандарті RS232 називається "space", йому відповідає рівень сигналу +5 .. +15 В.

При відсутності передачі на лінії завжди логічна одиниця. Про початок передачі даних сигналізує старт біт(Start)(логічний нуль). Кінець передачі даних сигналізує стоп біт (Stop)(логічна одиниця). Якщо під час прийому стоп біта на лінії буде логічний нуль, то приймач виставить прапор помилки прийому.

Тривалість одного біта у мікросекундах обчислюється за формулою: T = 10^6 / V, де V - швидкість передачі в бодах. Наприклад, для швидкості 9600 бод тривалість одного біта складає 1000000/9600 = 104 мкс.

Примітка К*онтролер працює з TTL рівнями сигналів, які відрізняються від рівнів інтерфейсу RS-232, тому для зв'язку комп'ютера з контролером по UART необхідно використовувати перетворювач рівнів RS-232 в TTL. (Для зв'язку двох контролерів перетворювачі можна не використовувати.) Для такого перетворення зазвичай використовується мікросхема MAX232.*

Схема підключення програмного уарта зображена на рисунку 3. Та схема включення датчика на

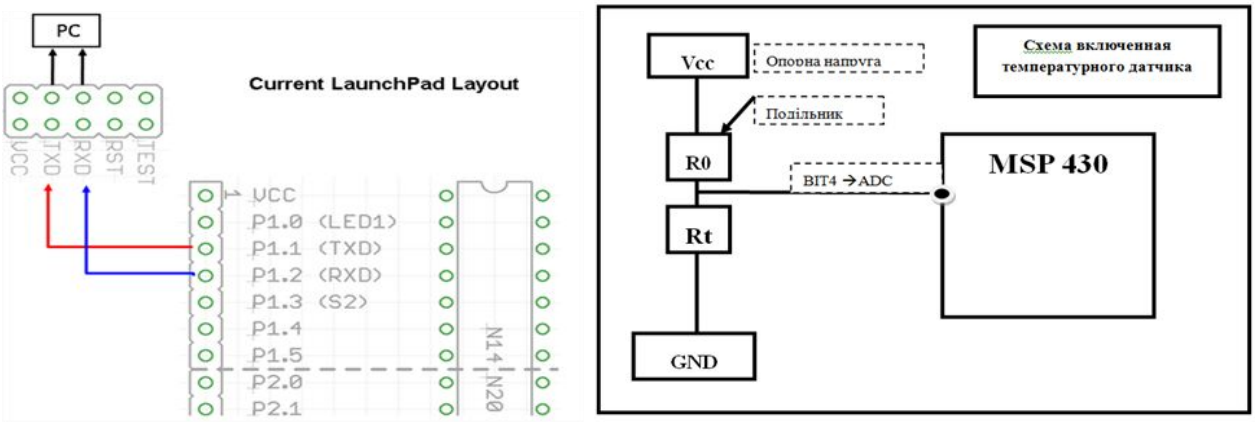


Рисунок 3 Використання програмного UART та Схема включення датчика

Блок схема роботи програми

Лістинг

Програма для мікроконтролера

```
#include "msp430g2231.h"
#define UART_TXD 0x02                    // TXD on P1.1 (Timer0_A.OUT0)..0x02 --> BIT1
#define A4        BIT4                   // Вхід для АЦП
#define CONVERSION_VALUE 0.003222
#define UART_TBIT (1000000 / 9600)       // 9600 Baud, SMCLK = 1MHz
                                         // Globals for transmit UART communication
unsigned int txData; // UART internal variable for TX
//-------------------------Function//------------------------------
//Оголошення функцій
void Init_Watchdog_1MHz(void);
void Iniy_Pin(void);
void Init_TaimerA_For_UART_transmit(void);
void TimerA_UART_tx(unsigned char byte);
void TimerA_UART_print(char *string);
void ADC_init(void);
void itoa(unsigned int val, char *str, unsigned int limit);
//-------------------------Main---------------------------------- */
void main(void){
  int adcValue; //Змінна для збереження отриманого коду з АЦП
                //Ініціалізація функ-й
   Init_Watchdog_1MHz();
   Iniy_Pin();
   Init_TaimerA_For_UART_transmit();
   ADC_init();

   _BIS_SR(GIE);                         // Enable CPU interrupts
    for(;;){
    P1DIR    |= BIT0 + BIT6;
    P1OUT ^=    BIT6+ BIT0;
 //     __delay_cycles(500000);    затримка 1ск
          //дозволяємо перетворення та вибірку
      ADC10CTL0 |= ENC + ADC10SC;
          //ENC -    перетвореення
          //ADC10SC - програмний запуск/зупика процесу виборки
          // очікуємо доки закінчеться перетворення
      while ((ADC10CTL1 & ADC10BUSY) == 0x01);
      adcValue=ADC10MEM;    // зберігаємо результат
          //забороняємо перетворення та вибірку
      ADC10CTL0 ^= ENC + ADC10SC;
  //--------------convert Int--------------------
          //ковертуємо результат    Int--char
   char itoad[16];
   itoa(adcValue,itoad,3000);
```

```
   //---------------send int---------------------
            //посилаємо значення по ком порту
   TimerA_UART_print(itoad);
            //щоб числа не прийшли у вигля потяга..розділимо їх таким посилом
   TimerA_UART_print("\r\n");
            //затримка 0,25ск
  __delay_cycles(250000);
}}
//--------------------------Functiuo----------------------------------
//Реалізація ф-ї itoa оскільки IAR не має деяких бібліотек С
void    itoa(unsigned int val, char *str, unsigned int limit){
   int temploc = 0;
   int digit = 0;
   int strloc = 0;
   char tempstr[5]; //16-bit number can be at most 5 ASCII digits;
    if(val>limit)
      val %= limit;
    do {
      digit = val % 10;
      tempstr[temploc++] = digit + '0';
      val /= 10;
   } while (val > 0);
    // reverse the digits back into the output string
   while(temploc>0)
      str[strloc++] = tempstr[--temploc];
      str[strloc]=0;
}
//===========================================================================*/

void    ADC_init(void) {
      //Регістр управління 1-го модуля АЦП
      ADC10CTL1 = INCH_4 + ADC10DIV_3 ;//+ CONSEQ_1;
            // INCH_4 - вибір вхідного каналу A4 --P1.4--BIT4
            // ADC10DIV_3 - Коефіцієнт ділення тактового сигналу..010-3
      //Регістр управління 0-го могдуля АЦП
     ADC10CTL0 = SREF_0 + ADC10SHT_3 + REFON + ADC10ON;// Ref voltage/sample&hold
            // SREF_0 - Джерело опорної напруги Vr+=Vcc.Vr-=Vss
            // ADC10SHT_3 - Час виборки 64такта ADC10CLK
            // REFON - Вмикання генератора опорної напруги
            // ADC10ON - Вмикання модуля ADC10
   } // ADC_init*/
Void    Init_Watchdog_1MHz(void){
         //Зупинка сторожового таймера
   WDTCTL = WDTPW + WDTHOLD;                                    // Stop watchdog timer
         //Налашування частоти роботи контролера 1.1MHZ(Є стандартно)
   DCOCTL = 0x00;                                               // Set DCOCLK to 1MHz
```

```
   BCSCTL1 = CALBC1_1MHZ;
   DCOCTL = CALDCO_1MHZ;
}
void Iniy_Pin(void){
      //налаштування портів
   P1OUT = UART_TXD;                                    // Initialize P1.1
   P1SEL = UART_TXD;                                    // Timer function for TXD pin
   P1DIR = UART_TXD;                                    // Set TXD pin to output
}
void Init_TaimerA_For_UART_transmit(void){
          //TA0CCTLx - регістр управлінн блоком захвата зрівняння
      TA0CCTL0 = OUT; // Timer_A for transmit UART operation
         // OUT - стан виходу Високий/низький рівень
         // Set TXD Idle as Mark = '1'
   TA0CCTL1 = SCS + CM1 + CAP;      // Sync, Neg Edge, Capture
         /* SCS - Синхронізація захоплення з тактовим сигналом таймера
         Асинхронний/Синхронний    */
         // CM1 - Зежим захоплення 1- Захоплення по наростаючому фронту
         // CAP - Режим роботи блоку захвата 0/ зрівняння 1
      // TA0CTL - Регістр управління таймером А
      TA0CTL = TASSEL_2 + MC_2;                  // SMCLK, start in continuous mode
            // TASSEL_2 - Джерело тактового сигналу 2- SMCLK
            // MC_2 - Режим Таймера рахує до і від)
}
void TimerA_UART_tx(unsigned char byte){//Outputs one byte using theTimer_AUART
   while (TACCTL0 & CCIE);                              // Ensure last char got TX'd
   TA0CCR0 = TAR;                                       // Current state of TA counter
         // TA0CCR0 - регістр захвата /зрівняння
   TA0CCR0 += UART_TBIT;                                // One bit time till first bit
   //Добавляємо старт і стоп біти(в байт для відправки)
   txData = byte;                                 // Load transmit data, e.g. 'A'=01000001
   txData |= 0x100;                                    // Add mark stop bit, e.g. 101000001
   txData <<= 1;                                      // Add space start bit, e.g. 1010000010
   TA0CCTL0 = OUTMOD0 + CCIE;                        //Set TXD on, enable counter interrupt
         // OUTMOD0 - стан біту OUT
        //    CCIE - дозвіл переривань захват/зрівняння 1-дозволено 0-заборонено
}
//Функція передачі даних по програмному уарту
void TimerA_UART_print(char *string) {// Prints a string using the Timer_A UART
   while (*string)
      TimerA_UART_tx(*string++);
  }
//---------------------------------Interrupt-----------------------------------------
#pragma vector = TIMER0_A0_VECTOR                       // Timer_A UART - Transmit ISR
    __interrupt void Timer_A0_ISR(void) {
   static unsigned char txBitCnt = 10; // Лічильник бітів передачі
```

```
  TA0CCR0 += UART_TBIT;                          // Add Offset to CCRx
  if (txBitCnt == 0) {                           // All bits TXed?
    TA0CCTL0 &= ~CCIE;                           // All bits TXed, disable interrupt
    txBitCnt = 10; //8+2 (байт+старт+стоп біт)   // Re-load bit counter
  }
  else {
    if (txData & 0x01) // біт = 1
      TA0CCTL0 &= ~OUTMOD2;                      // TX Mark '1'
    Else    // біт = 0
      TA0CCTL0 |= OUTMOD2;                       // TX Space '0'
  }
  txData >>= 1;// здвигаємо регістр даних в право    //Shift right 1 bit (low bits TX'ed first)
  txBitCnt--; // зменшуємо лічильник бітів
}
```

Коротко пояснення роботи программного уарту

/*Timer A interrupts (переривання таймера А..принцип відправки)

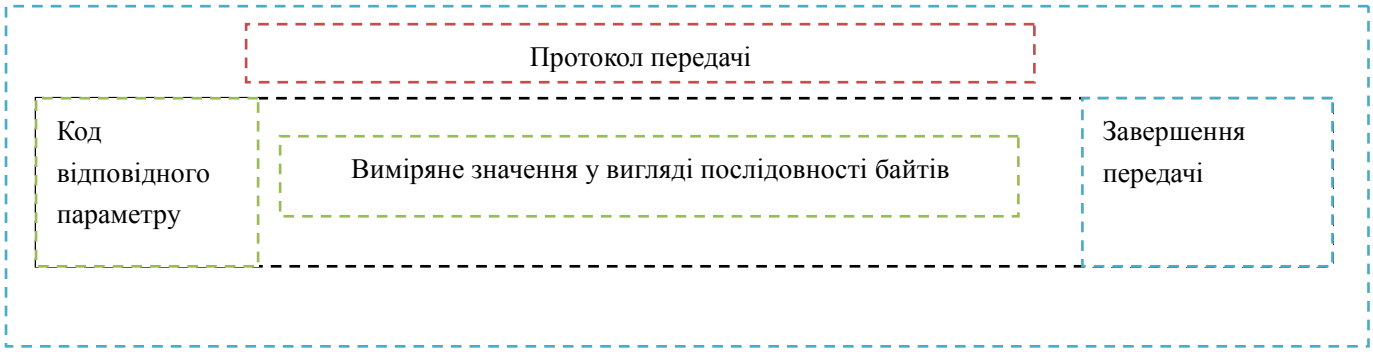
if 10 TXD біти послані ,відключити лічильник переривань TA0CCTL0 &= ~CCIE;

else

If low txData bit = 1 — transmit '1'; **else** — transmit '0'; Shift TXD right 1 bit

if (txData & 0x01) — TA0CCTL0 &= ~OUTMOD2; **else** — TA0CCTL0 |= OUTMOD2; txData >>= 1;*/

Протоколу передачі



Код відповідного параметру означає ключовий символ/слово яке відповідає за тип посланих даних.

У зв’язку з тим, що підчас виконання булла проблема з усіма потрібними датчиками використовуємо симуляцію для відправки даних з контролера н персональний комп’ютер.

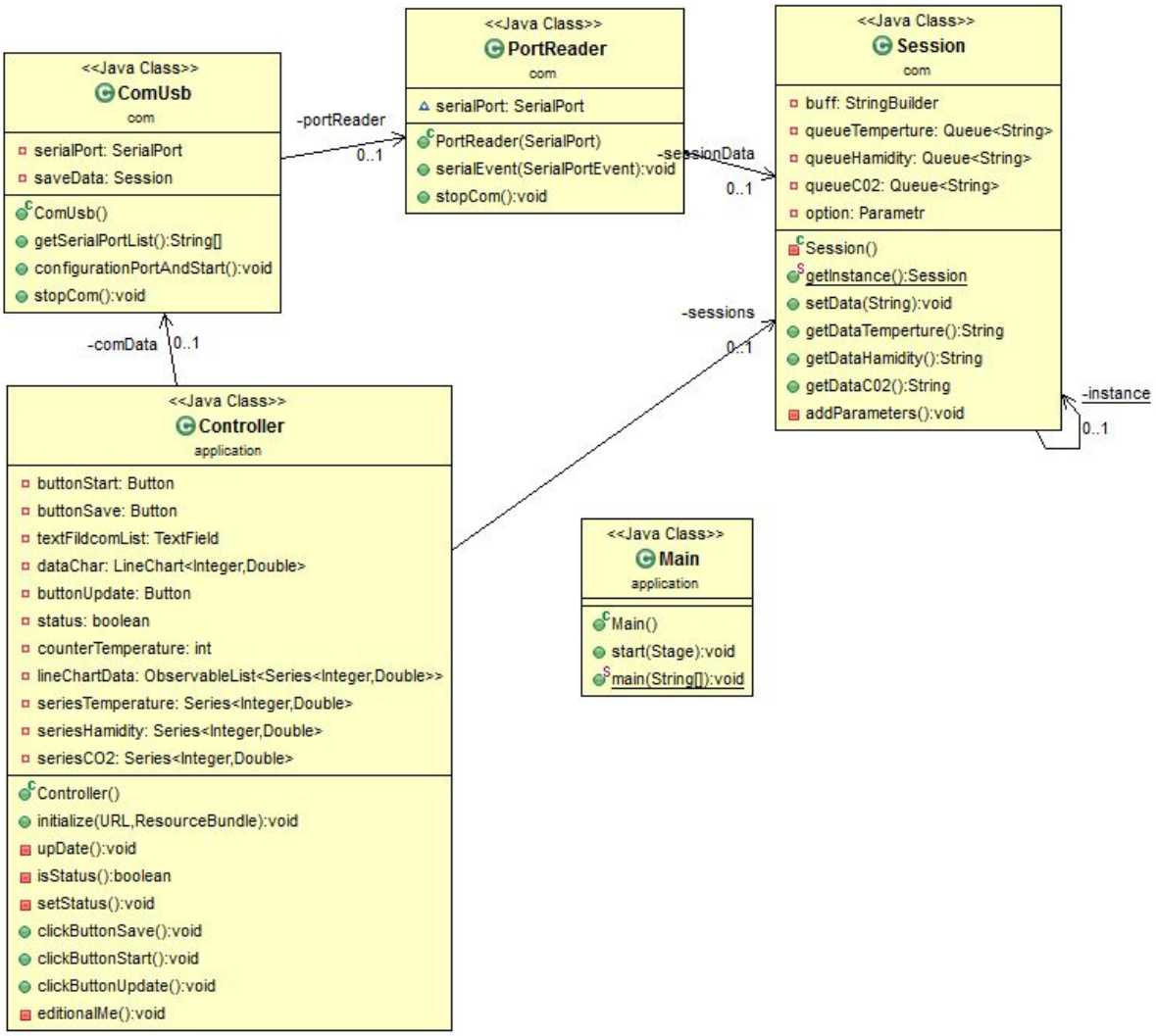
А саме додаємо в програму наступні частини коду відповідно до розробленого протоколу передачі.

```
TimerA_UART_print("s");
    char itoad[16];
```

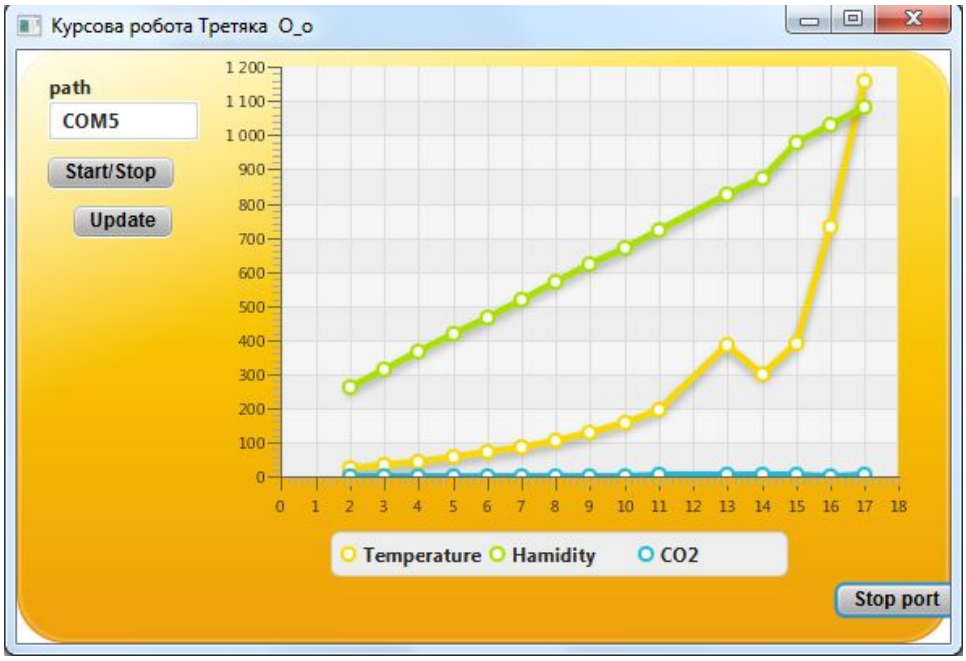
```
itoa( d2,itoad,64500);
TimerA_UART_print(itoad);                // Відправка температури
TimerA_UART_print("e");

TimerA_UART_print("w");
char itoadW[16];
itoa( d2,itoadW,64500);
TimerA_UART_print(itoadW);               // Відпрака вологості
TimerA_UART_print("e");

TimerA_UART_print("c");
char itoadC[16];
itoa( d2,itoadC,64500);
TimerA_UART_print(itoadC);               // Відправка рівня вуглекислого газу
TimerA_UART_print("e");
```

Реалізація роботи з ком портом

UML діаграма програми



Скріншот роботи програми. Графічна частина написана на JavaFx з використанням fxml.це є далеко не остаточний варіант, скоро вийде новіша версія і всі матеріали будуть на Github ,відповідно.



Налаштування Com порта. Class **ComUsb**

```
public void configurationPortAndStart()
{      //Передаём в конструктор имя порта
       serialPort = new SerialPort("COM5");
       try {
             //Открываем порт
             serialPort.openPort();
             //Выставляем параметры
    // serialPort.setParams(BAUDRATE_9600,DATABITS_8,STOPBITS_1,STOPBITS_1);//Set params
             serialPort.setParams(SerialPort.BAUDRATE_9600,
                                  SerialPort.DATABITS_8,
                                  SerialPort.STOPBITS_1,
                                  SerialPort.PARITY_NONE);
             //Включаем аппаратное управление потоком
             portReader = new PortReader(serialPort);
             serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_RTSCTS_IN |
                                           SerialPort.FLOWCONTROL_RTSCTS_OUT);
             //Устанавливаем ивент лисенер и маску
             serialPort.addEventListener(portReader, SerialPort.MASK_RXCHAR);
             //Отправляем запрос устройству
             //serialPort.writeString("Get data");
            // serialPort.closePort();
       }
       catch (SerialPortException ex) {
             System.out.println(ex);
```

Зчитування інформації з поту class **PortReader**

```
public class PortReader implements SerialPortEventListener {
    //Сесія для запису даних
    private Session sessionData = Session.getInstance();
    SerialPort serialPort;
    public PortReader(SerialPort serialPort){
        this.serialPort = serialPort;
    }
    @Override
    public void serialEvent(SerialPortEvent event) {
        if(event.isRXCHAR() && event.getEventValue() > 0){
            try {
             //Получаем ответ от устройства, обрабатываем данные и т.д.
             //byte[] buffer = serialPort.readBytes(10);
             String data = serialPort.readString(event.getEventValue());
                 sessionData.setData(data);
            }
            catch (SerialPortException    ex) {
                System.out.println(ex);
```

Запис даних .Class **Session**

```
public synchronized void setData(String str)    {
        if(str.contains("s")){ buff = new StringBuilder();    option = Parametr.TEMPERATURE; }
        else if(str.contains("w")){ buff = new StringBuilder();    option = Parametr.HAMIDY; }
        else if(str.contains("c")){ buff = new StringBuilder();    option = Parametr.CO2; }
        else if(str.contains("e")){
             addParameters();
             option = Parametr.NON;
        }
        else buff.append(str);
    }
```

Розкидаємо отримані дані відповідно з протоколом передачі

```
private void addParameters(){
                    if(option == Parametr.TEMPERATURE){
                    buff.append(" ");
                    String str2 = buff.toString();
                    queueTemperture.offer(str2);
                    }
                if(option == Parametr.CO2){
                    buff.append(" ");
                    String str2 = buff.toString();
                    queueC02.offer(str2);
                                        }
                if(option == Parametr.HAMIDY){
                    buff.append(" ");
                    String str2 = buff.toString();
```

```
                queueHamidity.offer(str2);

                }
            if(option == Parametr.NON){
```

Відповідно перелік можливих отриманих даних

```
public enum Parametr {
    TEMPERATURE,
    CO2,
    HAMIDY,
    NON }
```

Графічний інтерфейс та обробка отриманих даних

Створення/налаштування залежностей для відображення інформації на графіку

```
public class Controller implements Initializable {
private ObservableList<XYChart.Series<Integer, Double>> lineChartData = FXCollections.observableArrayList();
    private LineChart.Series<Integer, Double> seriesTemperature = new LineChart.Series<Integer, Double>();
    private LineChart.Series<Integer, Double> seriesHamidity = new
LineChart.Series<Integer, Double>();
    private LineChart.Series<Integer, Double> seriesCO2 = new
 LineChart.Series<Integer, Double>();
@Override
    public void initialize(URL location, ResourceBundle resources) {
              seriesTemperature.setName("Temperature");
              seriesHamidity.setName("Hamidity");
              seriesCO2.setName("CO2");
             lineChartData.addAll(seriesTemperature ,seriesHamidity,seriesCO2);
            dataChar.setData(lineChartData);
            dataChar.isResizable();
```

Приклад добавляння інформації на графік, відповідно

```
if(strTemperatur != null) seriesTemperature.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strTemperatur)));
if(strHamidity != null) seriesHamidity.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strHamidity)));
 if(strCO2 != null) seriesCO2.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strCO2)))
```

Весь лістинг програми приведено в Додатку 1.

Висновки

В ході виконнання даної роботи було реалізовано(вимірювальний канал) роботу з датчиком та обробку інформації з допомогою контролера та подальше передання інформації на персональний ПК де були побудовані графіки та відображена відповідна інформація з використанням java і відповідного пакету для роботи з ком портом jSSC.

Література

1. http://we.easyelectronics.ru/
2. http://www.msp430launchpad.com/
3. http://radiohlam.ru/
4. http://habrahabr.ru/
5. MSP430 Microcontroller Basics John H. Davies
6. http://spec-zone.ru/RU/Java/Docs/7/api/overview-summary.html
7. http://spec-zone.ru/RU/Java/Docs/7/api/overview-summary.html
8. https://code.google.com/p/java-simple-serial-connector/

Додаток 1

Листінг

```
public    class ComUsb {
      private SerialPort serialPort;
      private PortReader portReader;
      @SuppressWarnings("unused")
      private Session saveData = Session.getInstance();
      public ComUsb() {
      }
      public String[] getSerialPortList() {
            String[] portName = SerialPortList.getPortNames();
            return portName;
      }
      public void configurationPortAndStart()
      {      //Передаём в конструктор имя порта
            serialPort = new SerialPort("COM5");
            System.out.println("COM5");
            try {
                  //Открываем порт
                  serialPort.openPort();
                  //Выставляем параметры
                  serialPort.setParams(SerialPort.BAUDRATE_9600,
                                       SerialPort.DATABITS_8,
                                       SerialPort.STOPBITS_1,
                                       SerialPort.PARITY_NONE);
                  //Включаем аппаратное управление потоком
                  portReader = new PortReader(serialPort);
                  serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_RTSCTS_IN |
                                                SerialPort.FLOWCONTROL_RTSCTS_OUT);
                  //Устанавливаем ивент лисенер и маску
                  serialPort.addEventListener(portReader, SerialPort.MASK_RXCHAR);
                  //Отправляем запрос устройству
                  //serialPort.writeString("Get data");
                 // serialPort.closePort();
            }
            catch (SerialPortException ex) {
                  System.out.println(ex);
            }
      }
      public void stopCom() throws Exception
      {
            try {
                  //serialPort.closePort();
                  if(portReader != null)
```

```
                portReader.stopCom();
            } catch (SerialPortException    e) {
                // TODO Auto-generated catch block
                throw new Exception("Problem with port" + e.getMessage());
            }
        }
}
```

```
package com;
public class PortReader implements SerialPortEventListener {
    private Session sessionData = Session.getInstance();
    SerialPort serialPort;
    public PortReader(SerialPort serialPort){
        this.serialPort = serialPort;
    }
    @Override
    public void serialEvent(SerialPortEvent event) {
        if(event.isRXCHAR() && event.getEventValue() > 0){
            try {
             //Получаем ответ от устройства, обрабатываем данные и т.д.
             //byte[] buffer = serialPort.readBytes(10);
             String data = serialPort.readString(event.getEventValue());
              sessionData.setData(data);
              }
            catch (SerialPortException    ex) {
                System.out.println(ex);
            }
        }
    }/*serialEvent*/
    public void stopCom() throws SerialPortException
    {
        try {
            serialPort.closePort();
        } catch (SerialPortException e) {
            // TODO Auto-generated catch block
            throw new SerialPortException(null, null, null);
        }
    }
```

```
package enums;
public enum Parametr {
    TEMPERATURE,
    CO2,
    HAMIDY,
    NON }
```

```
package com;
import enums.Parametr;
public class Session{
private volatile static    Session instance ;
    private Session(){
    }
    public static Session getInstance(){
        if(instance == null){
            synchronized (Session.class) {
                if(instance == null) instance = new Session();
            }
        }
        return instance; // or Singleton.instance
    }
    /*Main Program*/
    private StringBuilder buff = new StringBuilder();
    private Queue<String> queueTemperture =    new LinkedList<String>();
    private Queue<String> queueHamidity =    new LinkedList<String>();
    private Queue<String> queueC02 =    new LinkedList<String>();
    private Parametr option = Parametr.NON;
    public synchronized void setData(String str)    {
    if(str.contains("s"))    { buff = new StringBuilder();    option = Parametr.TEMPERATURE; }
    else if(str.contains("w")) { buff = new StringBuilder();    option = Parametr.HAMIDY; }
    else if(str.contains("c")) { buff = new StringBuilder();    option = Parametr.CO2; }


    else if(str.contains("e")){
        addParameters();
            option = Parametr.NON;
        }
        else buff.append(str);
    }
    public    String getDataTemperture() throws NoSuchElementException{
         if (queueTemperture.size() == 0) throw new NoSuchElementException();
        return queueTemperture.remove();
    }
    public    String getDataHamidity() throws NoSuchElementException{
         if (queueHamidity.size() == 0) throw new NoSuchElementException();
        return queueHamidity.remove();
    }
    public    String getDataC02() throws NoSuchElementException{
         if (queueC02.size() == 0) throw new NoSuchElementException();
        return queueC02.remove();
    }
```

```
    private void addParameters(){

                if(option == Parametr.TEMPERATURE){
                    buff.append(" ");
                    String str2 = buff.toString();
                    queueTemperture.offer(str2);
                    System.out.println(str2 + "T");
                    }
                if(option == Parametr.CO2){
                    buff.append(" ");
                    String str2 = buff.toString();
                    queueC02.offer(str2);
                    System.out.println(str2 + "C");
                    }
                if(option == Parametr.HAMIDY){
                    buff.append(" ");
                    String str2 = buff.toString();
                    queueHamidity.offer(str2);
                    System.out.println(str2 + "H");
                    }
                if(option == Parametr.NON){
```

```
package application;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            Parent root = FXMLLoader.load(getClass().getResource("Interfaces.fxml"));
            Scene scene = new Scene(root, primaryStage.getWidth(), primaryStage.getHeight());
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setTitle("Курсова робота Третяка    O_o");
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {

        launch(args);
    }
}
```

```
package application;
public class Controller implements Initializable {
```

```
    @FXML private Button buttonStart;
    @FXML private Button buttonSave;
    @FXML private TextField textFildcomList;
    @FXML private LineChart<Integer, Double> dataChar;
    @FXML private Button buttonUpdate;
    private boolean status = true;
    private int counterTemperature = 0;
    private Session sessions = Session.getInstance();
    private ComUsb comData = new ComUsb();
    private ObservableList<XYChart.Series<Integer, Double>> lineChartData = FXCollections.observableArrayList();
    private LineChart.Series<Integer, Double> seriesTemperature = new LineChart.Series<Integer, Double>();
    private LineChart.Series<Integer, Double> seriesHamidity = new
LineChart.Series<Integer, Double>();
    private LineChart.Series<Integer, Double> seriesCO2 = new
LineChart.Series<Integer, Double>();

    @Override
    public void initialize(URL location, ResourceBundle resources) {
            seriesTemperature.setName("Temperature");
            seriesHamidity.setName("Hamidity");
            seriesCO2.setName("CO2");

           lineChartData.addAll(seriesTemperature ,seriesHamidity,seriesCO2);
           dataChar.setData(lineChartData);
           dataChar.isResizable();

        }
    private synchronized void upDate(){
        try{
            while(isStatus()){
                counterTemperature++;
                String strTemperatur = Param.tempCount(sessions.getDataTemperture());
                String strHamidity =  sessions.getDataHamidity();
                String strCO2 = Param.tempCO2(sessions.getDataC02());
if(strTemperatur == null && strHamidity == null && strCO2 == null) {setStatus();}
if(strTemperatur != null) seriesTemperature.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strTemperatur)));
if(strHamidity != null) seriesHamidity.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strHamidity)));
if(strCO2 != null) seriesCO2.getData().add(new XYChart.Data<Integer, Double>(counterTemperature,
Double.parseDouble(strCO2)));
            }
            setStatus();
            }catch (NoSuchElementException | NumberFormatException e) {}
      }
    private boolean isStatus() {
```

```
        return status;
    }
        private void setStatus() {
        this.status = !status;
    }
    @FXML
    public void clickButtonSave(){ // must be clickButtonClear! sorry
        seriesTemperature.getData().clear();
        seriesHamidity.getData().clear();
        seriesCO2.getData().clear();
        try {
            comData.stopCom();
        } catch (Exception e) {

            e.printStackTrace();
        }
    }
    @FXML
    public void clickButtonStart(){
        Thread my = new Thread(new Runnable() { @Override public void run()
{comData.configurationPortAndStart();}});
        my.setDaemon(true);
        my.start();
        String[] comList = comData.getSerialPortList();
        String com = " ";
        if(comList.length !=0){
            for(int i=0 ; i < comList.length; i++){
                com += comList[i];
            }
            textFildcomList.setText(com);
        }
        upDate();
    }
    @FXML
    public void clickButtonUpdate(){
        editionalMe();
    }
    private void editionalMe(){
        upDate(); //:)
```

```
package parameter.counting;
public class Param {
    public static String tempCount(String str){
        double tmpN = Double.parseDouble(str);
        double rt = (3960)/(3.3-(tmpN*0.0032));
```

```
            tmpN = (rt - 1200)/(1200*0.00784);
        return    String.valueOf(tmpN);
        }
        public static String tempCO2(String str){
            double tmpN = Double.parseDouble(str);
            tmpN = (tmpN*5.22)/10; //mPPM
            return    String.valueOf(tmpN);
        }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.chart.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.control.Button?>
<AnchorPane fx:id="mainPanel" prefHeight="366.0" prefWidth="577.0" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.Controller" >
    <children>
        <Button fx:id="buttonStart" layoutX="19.0" layoutY="66.0" mnemonicParsing="false"
onAction="#clickButtonStart" text="Start/Stop" />
        <Button fx:id="buttonUpdate" layoutX="35.0" layoutY="96.0" mnemonicParsing="false"
onAction="#clickButtonUpdate" text="Update" />
        <Button fx:id="buttonSave" layoutX="507.0" layoutY="330.0" mnemonicParsing="false"
onAction="#clickButtonSave" text="Stop port" />
        <Label layoutX="20.0" layoutY="14.0" prefHeight="16.0" prefWidth="76.0" text="path" />
        <LineChart fx:id="dataChar" layoutX="112.0" layoutY="-5.0" prefHeight="335.0" prefWidth="447.0">
          <xAxis>
            <NumberAxis    fx:id="xAxis" />
          </xAxis>
          <yAxis>
            <NumberAxis    fx:id="yAxis" />
          </yAxis>
        </LineChart>
        <TextField fx:id="textFildcomList" layoutX="19.0" layoutY="31.0" prefHeight="25.0" prefWidth="94.0" />
    </children>
</AnchorPane>
```

CSS

```
#mainPanel{
-fx-background-color:
            linear-gradient(#ffd65b, #e68400),
            linear-gradient(#ffef84, #f2ba44),
            linear-gradient(#ffea6a, #efaa22),
            linear-gradient(#ffe657 0%, #f8c202 50%, #eea10b 100%),
            linear-gradient(from 0% 0% to 15% 50%, rgba(255,255,255,0.9), rgba(255,255,255,0));
        -fx-background-radius: 30;
```

```
    -fx-background-insets: 0,1,2,3,0;
    -fx-text-fill: #654b00;
    -fx-font-weight: bold;
}
.button {
    -fx-text-fill: black;
    -fx-font-family: "Arial";
    -fx-font-weight: bold;
}
```

Додаток 2 jSSC

Крім операцій читання / запису jSSC надає наступний ряд можливостей:

- Управління лініями RTS, DTR
- Отримання статусу ліній CTS, DSR, RING, RLSD
- Отримання кількості байт в буферах
- Очищення буферів порту
- Відправка сигналу Break
- управління потоком
- Отримання списку com-портів в системі

Силка для скачування.
https://code.google.com/p/java-simple-serial-connector/