НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних систем, мереж та кібербезпеки

_____ Касаткін Д.Ю., к.пед.н., доц.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

На тему: Розробка програмно-апаратного пристрою виміру швидкості вітру

Спеціальність 123 «Комп'ютерна інженерія»

Гарант освітньої програми

к.фіз.-мат.н., доц. _____ / Нікітенко Є.В. /

(підпис) (ПІБ)

Керівник дипломного проекту: / Нікітенко Є.В./

(підпис) (ПІБ)

Виконав: / Цілик М.В./

(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки

_____/ Касаткін Д.Ю, доцент, к.пед.н /

підпис

" " 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студент Цілик Микита В'ячеславович

Спеціальність 123 «Комп'ютерна інженерія»

Тема бакалаврської кваліфікаційної роботи: Розробка програмно-апаратного пристрою виміру швидкості вітру

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2250 "С"

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до роботи: опис програмного забезпечення

Перелік питань , які потрібно розробити:

Аналіз проблемної області, вибір та обґрунтування засобів для розробки системи, проектування інформаційної системи.

Дата видачі завдання "16" 12 2024 р.

Керівник бакалаврської кваліфікаційної роботи

к.фіз.-мат.н., доц. _____ Нікітенко Є.В

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Цілик М.В.

(підпис) (ПІБ студента)

1.1.1-1

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання бакалаврської кваліфікаційної роботи	Строк виконання етапів бакалаврської кваліфікаційної роботи	Примітка
	Отримання завдання бакалаврської кваліфікаційної роботи		
	Початок планування системи	23.01.2025 – 03.02.2025	
	Побудова інтерфейсу		
	Розробка програми і тестування		
	Написання пояснювальної записки		
	Перевірка на плагіат		
	Відправка кваліфікаційної записки		
	Захист кваліфікаційної роботи		

Студент _____ Цілик О.С.

Керівник бакалаврської кваліфікаційної роботи

_____ Нікітенко Є.В.

РЕФЕРАТ

Об’єктом розробки є програмно-апаратний засіб виміру швидкості вітру.

Метою теоретичної частини роботи є дослідження існуючих засобів виміру швидкості вітру та їх аналізу. Метою проектувальної частини є опис принципу дії, побудова принципових та структурних схем. Практична частина полягає в створенні програмного та апаратного засобу виміру швидкості вітру, а саме створення невеликої метеостанції, здатної на перерахунок, зберігання, аналіз, прогнозування метеообстановки і отримання даних на персональний комп’ютер для користувача у зручному інтерфейсі.

Створений програмно-апаратний засіб може використовуватись в метеорології та кліматології.

Робота має практичну цінність.

Розрахунок економічної ефективності не проводився.

ПРОГРАМНО-АПАРАТНИЙ ЗАСІБ, КОНТРОЛЕР,
ІНТЕРФЕЙС ПРОГРАМИ, C, JAVA.

ABSTRACT

The object of development is a software and hardware tool for measuring wind speed.

The purpose of the theoretical part of the work is to study existing tools for measuring wind speed and their analysis. The purpose of the design part is to describe the principle of operation, build schematic and structural diagrams. The practical part consists in creating a software and hardware tool for measuring wind speed, namely, creating a small weather station capable of calculating, storing, analyzing, forecasting weather conditions and receiving data on a personal computer for the user in a convenient interface.

The created software and hardware tool can be used in meteorology and climatology.

The work has practical value.

The calculation of economic efficiency was not carried out.

SOFTWARE AND HARDWARE TOOL, CONTROLLER, INTERFACE PROGRAM, C, JAVA.

ЗМІСТ

ВСТУП 6

2 РОЗДІЛ АНАЛІЗ ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ ВИМІРУ ВІТРУ 9

2.1 АНАЛІЗ ІСНУЮЧИХ ЗАСОБІВ ВИМІРУ ВІТРУ 9

2.1.1 WeatherLink USB 6510 13

2.1.2 Портативний прилад TFA 426002 Vaavud 15

2.1.3 Цифровий анемометр AZ8919 17

2.1.4 Метеостанція La Crosse MA10050 18

2.1.5 Порівняння існуючих засобів виміру вітру 19

3 РОЗДІЛ ПРОЕКТУВАННЯ ПРОГРАМНОЇ ТА АПАРАТНОГО ЗАСОБУ ВИМІРУ ШВИДКОСІ ВІТРУ 21

3.1 ПРИНЦИП ДІЇ ПРОГРАМНО-АПАРАТНОГО ЗАСОБУ ВИМІРУ ШВИДКОСТІ ВІТРУ 21

3.2 Постановка завдання 22

3.3 Архітектура апаратної частини 23

3.4 Мікроконтролер ATMEGA8 27

3.4.1 Годинник реального часу DS1307 31

3.5 Архітектура програмної частини 34

4 РОЗДІЛ РЕАЛІЗАЦІЯ СИСТЕМИ 39

4.1 Вибір мови програмування для апаратної частини 39

4.2 Реалізація програмної частини 40

4.3 Вибір мови програмування для програмної частини 43

4.4 Реалізація програмної частини 44

5 ВИСНОВКИ 47

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 48

Додаток А....49

Додаток Б....50

Додаток В....57

ВСТУП

В сучасному житті безпека життєдіяльності стоїть на першому місці. Адже на сьогоднішній день людину підстерігає небезпека на кожному кроці. Під час виконання складних операцій на заводі чи будівництві ми можемо отримали значні ушкодження і деякі з них є смертельними. Також при активному відпочинку є велика ймовірність травмуватися.

За приклад можна взяти одну з небезпечних діяльностей людини, де найбільше зафіксовано травм та смертельних випадків – це будівництво. На будівельному майданчику однією з основних робіт є робота баштового крана. Найчастіше нещасні випадки стаються за його участі. Перед початком робіт кранівник має точно розрахувати свої дії і можливості техніки, для переміщення великогабаритного вантажу. В цьому випадку важливими перешкодами, в виконанні даних робіт, є погодні умови, а саме вітер. На допомогу будівельникам прийде прилад програмно-апаратний засіб виміру вітру - анемометр, який вимірює всі параметри вітру. Він встановлюється на стрілу, яка є найуразливіша частина баштового крану. Якщо погодні умови не дозволяють виконувати безпечно роботу, то необхідно перенести виконання, ніж підлягати ризику життя людей .

Людина не уявляє своє життя без технологій, на сьогодні вони її супроводжують всюди, починаючи з ранку до вечора. Технології не стоять на місці, а йдуть вперед стрімкими кроками, тому і дослідження погодних умов вже не в новинку. Існують різноманітні прилади для замірів швидкості вітру. Одні з них це програмно-апаратні.

Програмно-апаратний засіб виміру вітру, не тільки застосовується на підприємствах, будівництвах, і як вище згадано, використовується при активному відпочинку, так як і там людину взаємодіє з погодою і багато чого залежить від неї. Можемо розглянути яхтинг та парапланеризм, як приклади використання цього приладу. В цих видах відпочинку та спорті програмно-

апаратний засіб грає важливу роль. Виміри приладу дають змогу більш чітко і швидко виконувати безпечні маневри в повітрі і на воді. Завдяки йому людина може конкретніше оцінити ситуації та прийняти правильне рішення. В парапланеризмі пілот маневрує у повітрі більше часу, коли впевнений в своїх діях, так як знає швидкість, напрям і зміну потоків вітру в потрібний йому час.

Також не можна забувати, що програмно-апаратний засіб виміру вітру застосовується найчастіше у метеостанціях і є одним з основних приладів для моніторингу погодної ситуації у тому чи іншому регіоні країни. У країнах, де велика ймовірність торнадо чи урагану, ці прилади встановлюються у великій кількості на площинах для більш точного виміру вітру і попередження заздалегідь людей про наближення катаклізму.

При своїй різноманітності вони мають, як переваги, так і недоліки. До переваг відносять високу точність, ефективність, легкість в користуванні, виведення даних на ПК та можливість користуватися ними через Інтернет. Але й недоліки також присутні, не всі прилади мають можливість виводити свої дані на ПК та Інтернет, вони мають високу вартість, як програмної, так і апаратної частини приладу, відзначаються малою ефективністю, а в деяких – присутній незрозумілий інтерфейс.

Метою кваліфікаційної роботи є розробка програмно-апаратного засобу для виміру швидкості вітру, в якому максимально усунуті перераховані недоліки. Ціль – досягти високу ефективність за мінімальні затрати. Головною перевагою цього пристрою є те, що програмна частина не прикріплена до єдиної платформи, її можна встановлювати, як на Linux, Windows, так і MacOs. Це дає змогу не прив’язуватися до певної операційної системи, а працювати з тим, що зручно користувачеві.

На сьогодні прилад є невід’ємною частиною для людей, професії яких пов’язані з контрольно-вимірювальними пристроями, так як програмно-апаратний засіб є універсальним.

В майбутньому його можна модернізувати. При підключені програмно-апаратного засобу до ПК, дані одразу передаються в Інтернеті, висвітлюються

одразу на сайті та фіксується, за допомогою GPS, відповідно до місця знаходження.

1 РОЗДІЛ АНАЛІЗ ПРОГРАМНО-АПАРАТНИХ ЗАСОБІВ ВИМІРУ ВІТРУ

1.1 АНАЛІЗ ІСНУЮЧИХ ЗАСОБІВ ВИМІРУ ВІТРУ

Типовими приладами, безпосередньо призначеними для вимірювання швидкості вітру, є різноманітні анемометри, що використовують здатні до обертання чашечки або пропелери. Для вимірювання із більшою точністю, зокрема для наукових досліджень, використовують вимірювання швидкості звуку або вимірювання швидкості охолодження нагрітого дроту або мембрани під дією вітру. Іншим поширеним типом анемометрів є трубка Піто, що вимірює різницю динамічного тиску між двома концентричними трубками під дією вітру та широко використовується в авіаційній техніці.

Швидкість вітру на метеорологічних станціях більшості країн світу зазвичай вимірюють на висоті 10 м та усереднюють за 10 хвилин. Виняток становлять США, де швидкість усереднюють за 1 хвилину, та Індія, де її усереднюють за 3 хвилини. Період усереднення має важливе значення, оскільки, наприклад, швидкість постійного вітру, виміряна за 1 хвилину зазвичай на 14% вище значення, виміряного за 10 хвилин. Короткі періоди швидкого вітру досліджують окремо, а періоди, у які швидкість вітру перевищує усереднену за 10 хвилин швидкість щонайменше на 10 вузлів (5 м/с), називаються поривами. Шквалом називається подвоєння швидкості вітру, сильнішого за певний поріг, що триває хвилину або більше.

На території України розташовано близько 200 метеостанцій, підпорядкованих державній гідрометеорологічній службі (Державному комітету України по гідрометеорології, нині – Міністерству надзвичайних ситуацій), які входять до системи Всесвітньої метеорологічної асоціації (ВМО) – спеціалізованого міжурядового закладу ООН в області метеорології. На цих станціях виміри характеристик вітру виконуються за методикою ВМО. Інтервал зняття показів – 3 години, висота вимірювань – приблизно 10 м, округлення до

найближчих цілочисельних значень (в метрах за секунду). Такі покази можуть бути корисними при визначенні довготермінових властивостей вітрового потенціалу, проте для потреб вітроенергетики вони недостатні – ні за дискретністю вимірювань, ні за їх точністю, ні за висотою

Для дослідження швидкості вітрів у багатьох точках використовують зонди, швидкість яких визначають за допомогою GPS, радіонавігації або слідкування за зондом за допомогою радару або теодоліту. Іншими методами є використання таких методів, як содари, доплерівські лідари та радари, здатні вимірювати доплерівський зсув електромагнітного випромінювання, відбитого або розсіяного аерозольними частинками або навіть молекулами повітря. На додаток, радіометри і радари використовують повітря для вимірювання нерівності водної поверхні, що добре відображає приповерхневу швидкість вітру над океаном. За допомогою зйомки руху хмар з геостаціонарних супутників можна встановити швидкість вітру на більших висотах.

Швидкість вітру вимірюється метрами на секунду, кілометрами на секурду, вузлами, шкалою Бофорта, шкалою Фудзі та шкалою Сапфіра-Сімсона для ураганів.

Шкала Бофорта - дванадцятібальна шкала, прийнята Всесвітньою метеорологічною організацією для оцінки швидкості вітру по його дії на наземні предмети або за хвилювання у відкритому морі, приведено на таблиці 1.1. Середня швидкість вітру вказується на стандартній висоті 10 м над відкритою рівною поверхнею.

Шкала розроблена англійським адміралом Френсісом Бофортом в 1806 році. З 1874 року ухвалено для використання в міжнародній синоптичної практиці. Спочатку в ній не вказувалася швидкість вітру (додана в 1926 році). У 1955 році, щоб розрізняти ураганні вітри різної сили, Бюро погоди США розширило шкалу до 17 балів.

Варто відзначити, що висота хвиль в шкалі приведена для відкритого океану, а не для прибережної зони.

Таблиця 1.1 – Шкала Бофорта

Бал	Характеристика	м/сек	вузол	км/год
0	Безвітря	0,0...0,2	<1	<1
1	Легкий вітрець	0,3...1,5	1...3	1...5
2	Легкий бриз	1,6...3,3	4...6	6...11
3	М’який бриз	3,4...5,4	7...10	12...19
4	Помірний бриз	5,5...7,9	11...16	20...28
5	Свіжий бриз	8,0...10,7	17...21	29...38
6	Сильний бриз	10,8...13,8	22...27	39...49
7	Близький до штормового вітер	13,9...17,1	28...33	50...61
8	Штормовий вітер	17,2...20,7	34...40	62...74
9	Сильний штормовий вітер	20,8...24,4	41...47	75...88
10	Шторм	24,5...28,4	48...55	89...102
11	Надзвичайно сильний шторм	28,5...32,6	56...63	103...117
12	Ураган	>32,7	>64	>118

Напрямок вітру визначають щодо сторін світу і позначають або в румбах (8 або 16): північний, північно-східний, східний тощо, або в поділках: одна поділка містить 5^o або 10^o, залежно від необхідної точності вимірювань.

Поривчастість вітру - це стрибкоподібні підсилення і послаблення швидкості вітру.

Розглянемо повітряні потоки, що можуть викликати стресові ситуації.

Смерч - сильний вітер у вигляді горловини з вертикальною віссю, що має велику швидкість обертання. В Америці його називають торнадо. Причиною смерчу є великі градієнти тиску, нестійкість нижнього шару атмосфери (до 2 км), що виникає при зіткненні сухих холодних повітряних мас з теплими й вологими. Для оцінки смерчів використовують шкалу Фудзі наведену в таблиці 1.2.

Таблиця 1.2 – Шкала Фудзі

Категорія	Швидкість км/год	Наслідки
F0 (легкий)	64...115	Руйнування телевізійних антен, труб, дерев, вікон
F1 (помірний)	116...179	Перевертання автомобілів, виривання дерев з корінням
F2 (значний)	180...251	Знесення дахів, перевертання рухомих будинків
F3(суворий)	252...330	Руйнування металевих будівель, зсування зовнішніх стін, повали лісів та вилягання угідь
F4 (спустошливий)	331...416	Падіння стін, перенесення металевих та бетонних конструкцій на велику відстань
F5 (неймовірний)	417...509	Перенесення будинків на велику відстань, руйнування шкіл, мотелів
F6 (який важко уявити)	510...606	Автомобілі піднімає в повітря

Урагани - тропічні циклони, швидкість яких досягає 80 м/сек. Термін «ураган» стосується екстремальних вітрів, що виникають у північній Атлантиці; аналогічні явища у Тихому океані називають тайфунами. Тривалість ураганів від 1 до 30 днів. Оцінюють урагани за шкалою Сафіра-Сімсона, що наведена в таблиці 1.3.

Таблиця 1.3 – Шкала Сапфіра-Самсона оцінки ураганів

Категорія	Швидкість, км/год	Наслідки
1 (мінімальний)	119…153	Руйнування рухомих будинків, часткове затоплення прибережних районів

2 (помірний)	154…177	Суттєве пошкодження рослинності, виривання дерев, затоплення прибережних доріг
3 (великий)	178…279	Руйнування малих будинків, затоплення прибережної території на відстані до 13 км
4 (екстремальний)	210…249	Руйнування дахів, вікон, повне руйнування рухомих будинків, затоплення до 10 км
5 (катастрофічний)	>249	Руйнування будинків, промислових підприємств, потреба в евакуації населення в зоні 8…16 км

1.1.1 WeatherLink USB 6510

Для прикладу розглянемо Davis Instruments та інтерфейс для персонального комп'ютера WeatherLink USB 6510, який наведено на рис.1.3. Він призначений для дослідження погоди на професіональному рівні. Завдання цього приладу в тому, що на свій комп'ютер встановлюється ПЗ WeatherLink USB 6510 та реєстратор даних до будь-якої метеостанції Davis Instruments. Реєстратор даних з'єднуємо з консоллю Vantage Vue та Vantage Pro2 з модулем прийому передачі даних Weather Envoy чи Envoy8X. ПЗ WeatherLink USB 6510 дозволяє збирати та зберігати данні на ПК. Приєднавши реєстратор до комп'ютера, щоб збирати та завантажувати дані на жорсткий диск для більш детального аналізу та побудови графіку приведено на рисунку 1.1 та 1.2.

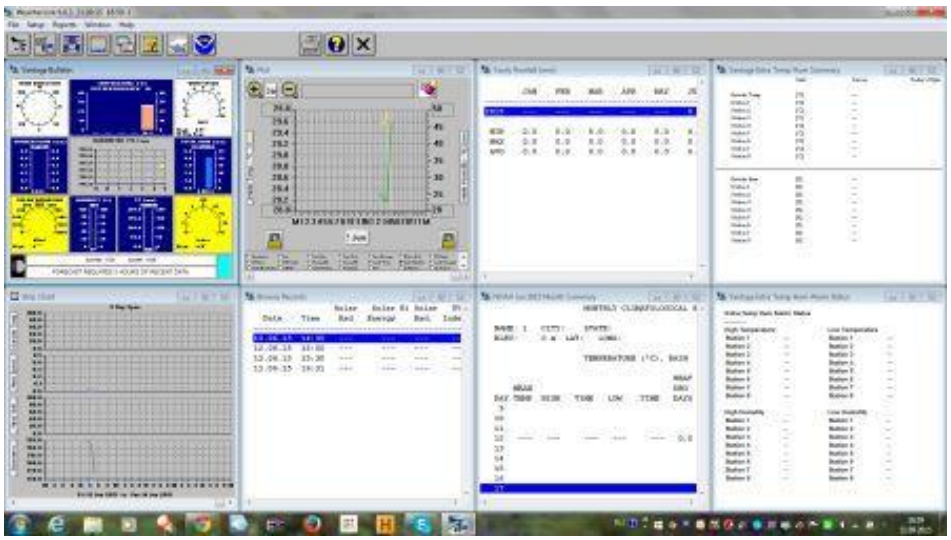


Рисунок 1.1 - Приклад збору даних на ПК

Розглянемо особливості програми WeatherLink USB 6510:

1) Розширена звітність;

2) Оброблення та аналіз даних, побудова графіків за день, тиждень, місяць, рік.

3) Порівняння даних. Дозволяє переглядати поточні погодні умови та переглядати декілька погодних змінних в той самій час для порівняння.

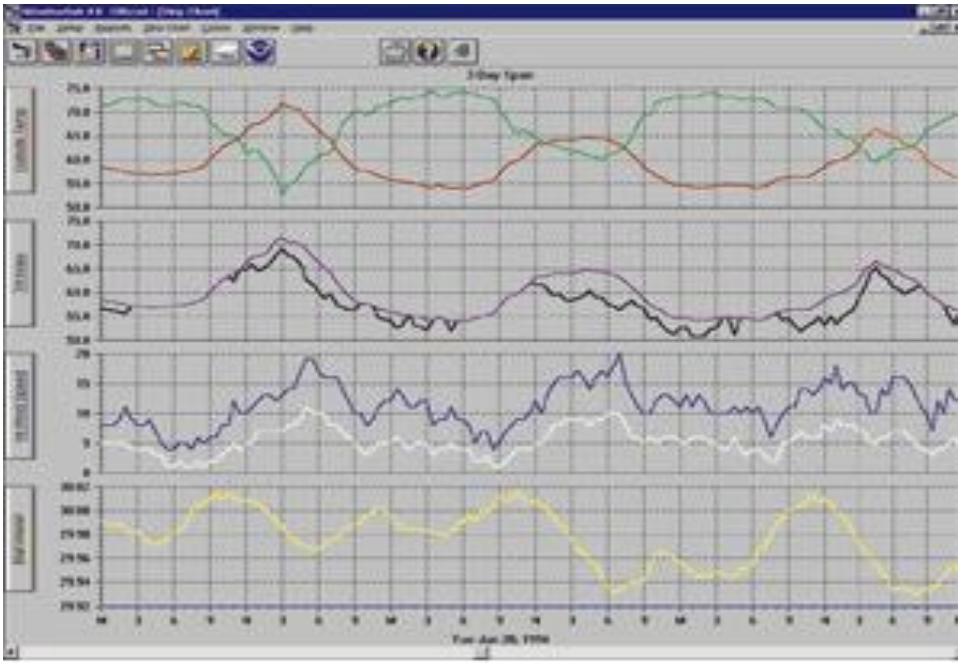


Рисунок 1.2 - Приклад побудови графіку

При підключенні додаткових датчиків сонячної радіації та УФ випромінювань проходить детальна інформація про ризики сонячної радіації.

Недоліками цього приладу є висока ціна, як на метеостанцію, так і на програмне забезпечення. До переваг можна віднести те, що цей програмно-апаратний засіб виміру вітру має багатофункціональність, ефективність, точність вимірювань, аналізує отримані дані за певний період часу, наочність побудованих графіків.

Рисунок 1.3 - Davis Instruments

1.1.2 Портативний прилад TFA 426002 Vaavud

Другим прикладом розглянемо портативний пристрій для виміру швидкості вітру обмін даних в онлайні – TFA 426002 Vaavud, що приведений на рисунку 1.4.

Прилад підключається до смартфону в гніздо для навушників. Для відображення даних необхідно завантажити додаток Vaavud. Чутливий елемент

при взаємодії з вітром відображає значення швидкості руху повітря на екрані смартфону.

Швидкість руху вітру передається в поточному часі, середнє та максимальне на дисплей смартфону. Ці значення дозволяють користувачу обирати в якій одиниці виміру необхідно отримати результат - у вузлах, кілометрах, метрах за секунду. Також є можливість передавати та завантажувати дані в Інтернет і переглядати значення даних для всіх користувачів в різних частинах світу.

Опорними функціями є:

-вимірювання швидкості вітру;

-вимірювання середнього і максимального значення;

-побудова графіка в режимі реального часу;

-точність виміру ±4% чи 0,25 м/с.

Недоліками цього пристрою є мала функціональність, не дозволяє користувачу обробляти отримані дані одразу на ПК, неточність вимірів. До переваг можна віднести легкість у використанні, ергономічність, можливість підключення до Інтернету, невелика вартість.

Рисунок 1.4 – Портативний прилад TFA 426002 Vaavud

1.1.3 Цифровий анемометр AZ8919

Наступним прикладом розглянемо цифровий крильчатий анемометр AZ8919, що приведений на рисунку 1.5. Він виявляє вологість повітря та швидкість вітру тільки в теперішньому часі і це його основний недолік. Якщо дослідити його функціональність, він є апаратним приладом, це також є суттєвим недоліком, який не дозволяє користувачу обробляти отримані дані, має малу точність вимірів.

Забезпечує вимірювання наступних параметрів:

-швидкість руху вітру;

-об’єм повітря;

-CO_2;

-Вологість;

-Температура;

-Точка роси.

К перевагам можна віднести тільки легкість у використанні, ергономічність та невелику вартість приладу, компактність.

Рисунок 1.5 - Крильчатий анемометр AZ8919

1.1.4 Метеостанція La Crosse MA10050

Наступним прикладом розглянемо метеостанцію La Crosse MA10050 приведену на рисунку 1.6, яка здатна передавати в Інтернет дані про температуру і відносну вологість повітря, швидкість і напрямок вітру, рівні опадів завдяки вбудованій технології Mobile-Alerts. Передану станцією інформацію ви можете оперативно переглядати на екрані вашого смартфона за допомогою попередньо встановленого програмного забезпечення. Завдяки наявності мобільного GSM-шлюза, користувач може підключити до 50 різних датчиків Mobile-Alerts. Великою перевагою є відображення історії показань метеостанції за останні 90 днів з можливістю вибору дати.

До переваг можна віднести ергономічність, багатофункціональність, ефективність, передавання отриманих даних в Інтернет. До недоліків можна віднести: велика вартість комплекту датчиків, неточність вимірювань, неможливість підключення до ПК і обробляти дані отримані зі смартфону.

Рисунок 1.6 - Метеостанцію La Crosse MA10050

1.1.5 Порівняння існуючих засобів виміру вітру

Дослідивши засоби виміру вітру дійшли до висновку, що існують безліч різноманітних приладів, які мають свої особливості.

Спостереження за вітром та виміри його параметрів показують, що швидкість і напрям вітру швидко змінюються, а величина змін залежить від часового інтервалу спостережень. Прилади мають різні характеристики та властивості. Це їх відрізняє один від одного. Тому користувачеві перед тим, як обрати той чи інший прилад, слід обрати, що саме він очікує від приладу, порівняти їх характеристики та обрати для себе необхідний. В таблиці 1.1 наведено порівняння деяких з існуючих видів приладів виміру вітру, за якою можна прослідкувати на скільки зручні, ефективні та функціональні у використанні всі наведені прилади. Характеристиками для порівняння обрано можливість підключення до ПК, виведення даних до Інтернету, портативність, легкість у користуванні та вартість.

Таблиця 1.1 – Порівняльна таблиця існуючих засобів виміру вітру

№ п/п	Назва приладу	Характеристики				
		Підключення до ПК	Виведення даних в Інтернет	Портативність	Легкість у використанні	Вартість
1.	WeatherLink USB 6510	+	+	-	-	-
2.	TFA426002Vaavud	+	+	-	+	+
3.	Крильчатий анемометр AZ8919	-	-	+	+	+
4.	Метеостанцію La Crosse MA10050	-	-	+	+	-

Проаналізувавши дані прилади можна зазначити прилади, які мають високу функціональність та можуть виводити дані в Інтернет є дорогими та складними у використанні, а свою чергу прилади з малою ефективністю мають помірну ціну.

ВИСНОВКИ

В розділі 1 проаналізували засоби виміру вітру. Існує безліч видів приладів. Вони відрізняються один від одного характеристиками, властивостями та ціною, але мають однакове призначення – вимірювання швидкості вітру.

Для виміру вітру використовується апаратні, програмні та програмно-апаратні прилади. Апаратні прилади розглянуті на основі цифрового анемометру AZ8919 та метеостанції La Cross Ma 10050. Програмно-апаратні прилади розглянуті на основі Davis Instruments та інтерфейсом ПК Weather Link USB 6510, портативного приладу TFA 42002 Vaavud.

Проаналізувавши наведені прилади, порівнявши їх характеристики, можна відзначити наступні переваги: ергономічність, багатофункціональність, ефективність, компактність, легкість у використанні. До недоліків слід віднести: вартість, неточність у вимірах, неможливість підключення до ПК та обробка даних, виведення їх до Інтернету. Для усунення цих недоліків запропоновано розробку даного програмно-апаратного засобу виміру швидкості вітру, який наведено в даній кваліфікаційній роботі.

2 РОЗДІЛ ПРОЕКТУВАННЯ ПРОГРАМНОЇ ТА АПАРАТНОГО ЗАСОБУ ВИМІРУ ШВИДКОСІ ВІТРУ

2.1 ПРИНЦИП ДІЇ ПРОГРАМНО-АПАРАТНОГО ЗАСОБУ ВИМІРУ ШВИДКОСТІ ВІТРУ

Принцип роботи програмно-апаратного засобу виміру вітру полягає: у виявленні зміни деякої фізичної властивості потоку за допомогою трьох елементів: чутливого елементу, апаратної частини, програмної частини. Структурна схема приведена на рис. 2.1. На рисунку 2.2 приведена блок-схема апаратної програми. На рисунку 2.3. наведено принципову схему апаратної частини.

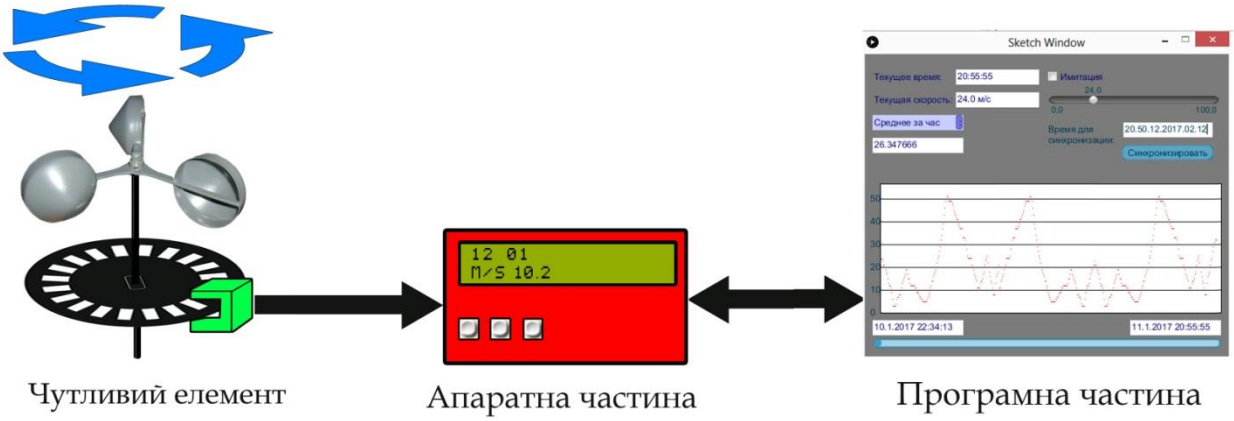


Рисунок 2.1 - Структурна схема програмно-апаратного засобу виміру вітру

1) Чутливий елемент складається з: чашковий елементу, перфорованого диску, які обертаються на рухомому валу, який обертається навколо своєї осі і оптичного датчику. Кожний отвір перфорованого диску є величина повороту чашковий елементу. Це значення фіксує оптичний датчик (у положенні отвору дорівнює одному, в положенні закритого оптичного дорівнює нулевому значенню). Таким чином, при фіксації куту повороту перфорованого диску, апаратна частина приймає двійкове значення.

2) Апаратна частина складається з: мікроконтролера ATmega8, LCD дисплей, таймер-годинник реального часу. Мікроконтролер ATmega8 э основною частиною апаратної частини, яка веде перерахунок двійкове значення з чутливого елементу і перераховує, фіксує, та передає значення на LCD дисплей, таймеру-годиннику реального часу, і також в програмну частину через COM-порт.

3) Програмна частина складається з: rx, tx блоків читання та передавання даних з COM-порту, блоку побудови графіка, блоку збереження даних з COM-порту, та блоку корегування часом для апаратної частини.

Апаратна частина також має багату функціональність, а саме за допомогою маніпуляторів корегувати час у годиннику реального часу, виробляти калібрування значення швидкості вітру і корегування часу прийнятого з персонального комп'ютера через інтерфейс програмної частини.

Програмна частина отримавши дані з COM-порту будує графік, зберігає данні в окремому файлі. Також програмна частина в режимі імітації в доступній формі може показати користувачу всі можливості програмної частини. А саме окремо від апаратної частини побудувати графік, зберігати данні також в тому ж окремому файлі, але данні для роботи будуть братися відносно реального часу на ПК та смуги прокрутки якою користувач самостійно корегує швидкість вітру.

2.2 Постановка завдання

Метою дипломної роботи є створення програмно-апаратного засобу виміру швидкості вітру, при розробці повинен володіти наступними особливостями:

1) Швидко та точно обчислювати кут оберту перфорованого диску оптичним датчиком;
2) Без похибки перетворювати дані з оптичного датчика, через тригер
3) Шмітта у двійкову систему числення;
4) Для користувачів повинна бути також реалізована можливість;

5) калібрування швидкості вітру та часу у апаратній частині;

5) Обчислювати після калібрування по новим перемінним;

6) Мати можливість виведення даних на LSD дисплей;

7) Мати можливість підключення апаратної частини до COM-порту;

8) Надавати можливість користувачу бачити дані з апаратної частини у програмному інтерфейсі;

9) Завжди зберігати дані виміру вітру з апаратної частини, в окремий файл на ПК;

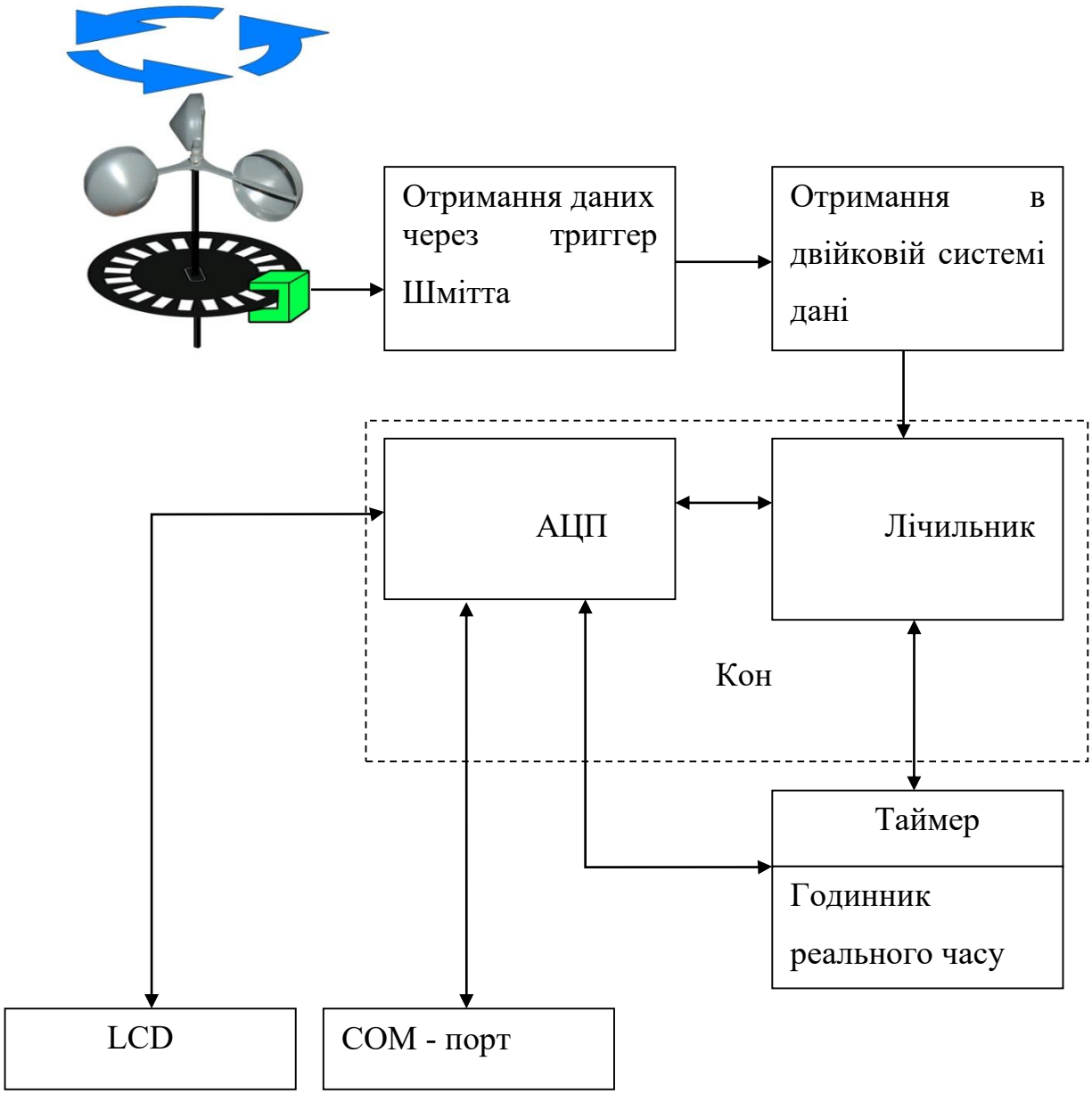
10) Надавати можливість тестування програмної частини;

11) Мати можливість отримання середнього значення вітру за годину, за 3 години та 6 годин;

2.3 Архітектура апаратної частини

Апаратна частина складається з: мікроконтролера ATmega8, LCD дисплей, таймер-годинник реального часу. Мікроконтролер ATmega8 э основною частиною апаратної частини, яка веде перерахунок двійкове значення з чутливого елементу і перераховує, фіксує, та передає значення на LCD дисплей, таймеру-годиннику реального часу, і також в програмну частину через COM-порт.

Дані в двійковій системі числення, які отримує оптичний датчик, приходять до контролера та визначаються таким чином: кожний отвір перфорованого диску є величина повороту чашкового елементу. У положенні отвору дорівнює одному, в положенні закритого оптичного дорівнює нулевому значенню). 40 отворів дорівнює одному оберту перфорованого диску. Таким чином при зміні куту повороту чутливого елементу, апаратна частина приймає двійкове значення через тригер Шмітта.



На рисунку 2.2 - Блок-схема апаратної програми

Тригер Шмітта - це компонент електронного пристрою, функція якого є формування постійно мінливого сигналу на вході в серію прямокутних імпульсів на виході. Застосовується в аналого-цифрових перетворювачів, фільтрах, лініях зв'язку. Тригер Шмітта має свою відмінність від інших видів тригерів тим, що він має єдиний вхід і один вихід і не має властивості пам'яті. Він складається з двох інверторів, що мають позитивно-зворотний зв'язок (ПЗС), в результаті чого стан виходу тригера може змінюватися лавиноподібно.

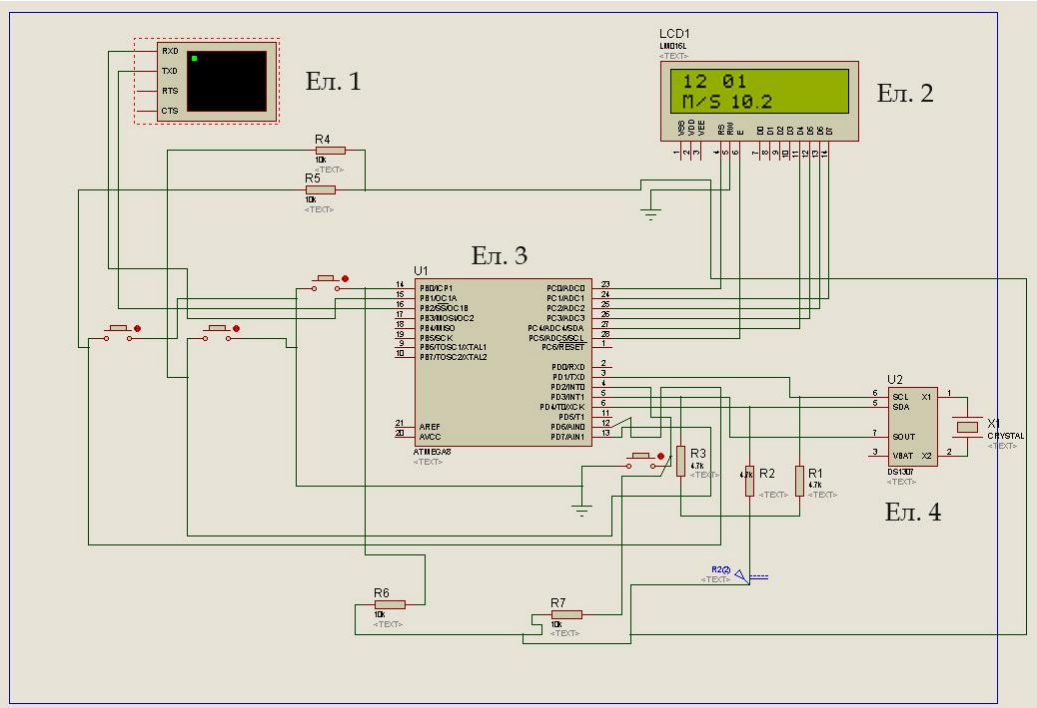


Рисунок 2.3- Принципова схема апаратної частини

Елемент 1. Віртуальний COM-порт, для моніторингу відправлених даних.

Елемент 2. LCD дисплей.

Елемент 3. Мікроконтролер ATmega8.

Елемент 4. Таймер-годинник реального часу.

Тригер Шмітта це компаратор, що має ПЗС. У даній схемі частка вихідного електричного сигналу ОУ надходить на прямий вхід і встановлює рівень, при якому схема буде переключатися.

Принципова схема роботи тригера Шмітта на ОУ приведена на рис. 2.4.

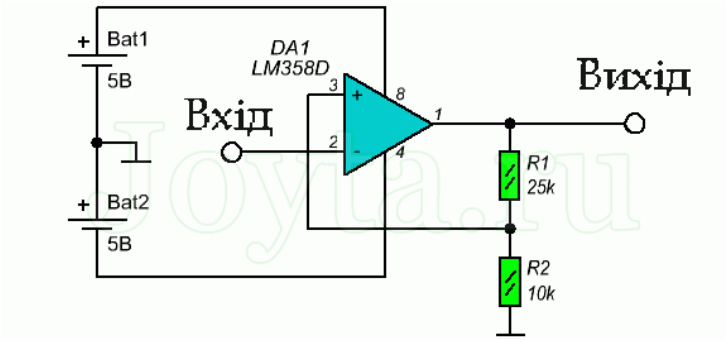


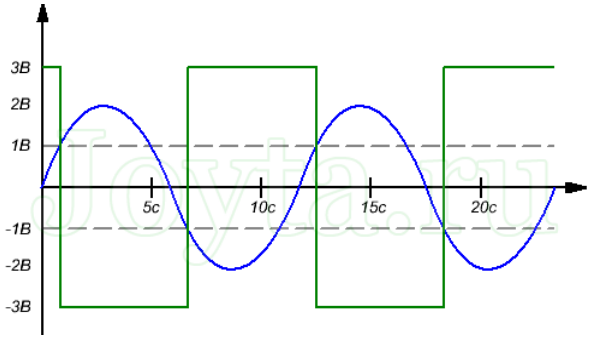
Рисунок 2.4 - Принципова схема тригера Шмітта

На рисунку 2.4. Принципова схема роботи тригера Шмітта з елементами: ОУ підключений до двополярного блоку харчування на 5 вольт. На інверсний вхід DA1 надходить синусоїдного сигнал рівний амплітуді 2 В. R1 і Опору R2 25 Мають значення і 10 кОм кОм. Напруга на прямому виведенні DA1 надходить з дільника напруги побудованого на резисторах R1 і R2, який підключений до виходу ОУ. Формула розрахунку для визначення напруги насичення:Uвх1 = +U*R2/(R1+R2) = 3,5*10/35 = 1 В

1. Uвх1 = -U*R2/(R1+R2) = -3,5*10/35 = -1 В

Коли на виході ОУ напруга з позитивним потенціалом насичення - на прямому вході напруга дорівнює 1 вольт. Припустимо, вхідний електричний сигнал поступово збільшується з нуля. Поки потенціал вхідного сигналу не перевищує напруги на прямому вході - схема знаходиться в стабільному стані. Тільки-но вхідний електричний сигнал перевершить величину в 1 вольт, напруга на вході ОУ змінить свою полярність на негативне напруга насичення. Це змінить напругу на прямому вході ОУ, і воно буде дорівнює -1 вольт.

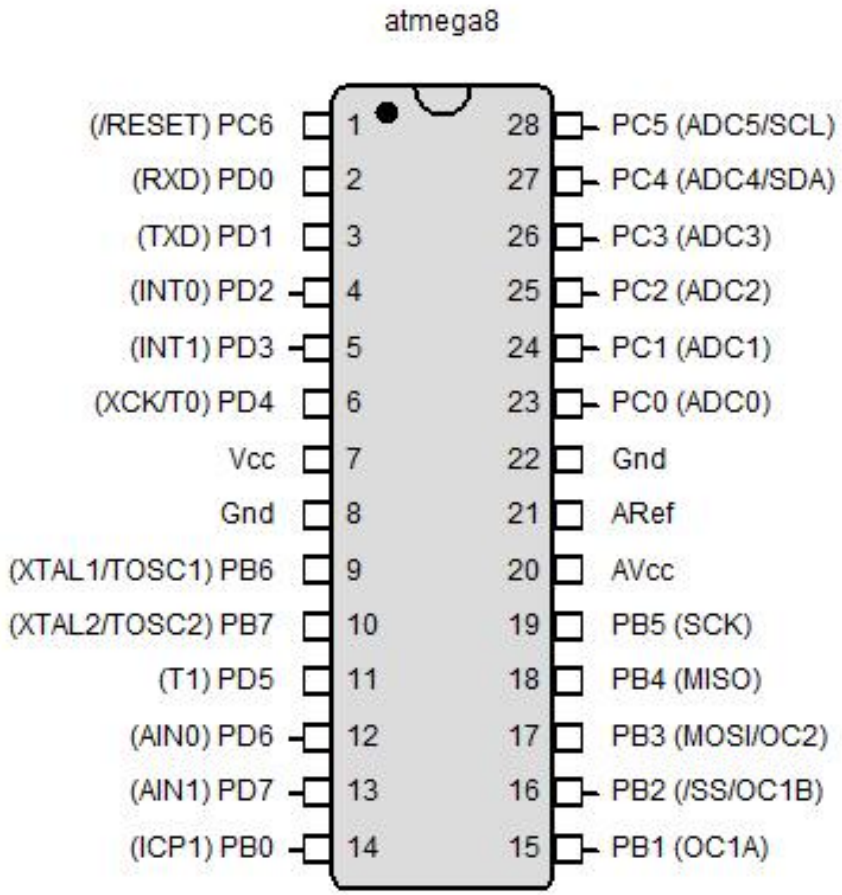
Вхідний електричний сигнал поступово буде збільшуватися до максимуму, а потім почне зменшуватися. Після того як амплітуда сигналу на вході стане менше 1 вольта, то на виході ОУ буде так само негативний потенціал насичення. Як тільки сигнал на вході пройде величину -1В, напруга на виході зміниться і буде рівним позитивного потенціалу насичення. На рисунку 2.5. можна спостерігати залежність вихідної напруги тригера Шмітта від вхідного. В результаті такої роботи схеми шуми вхідного сигналу не впливатимуть на вихідний сигнал.



Рисунку 2.5 - Графік залежність вихідної напруги від вхідного.

2.4 Мікроконтролер ATMEGA8

Мікроконтролер ATMEGA8 - мікросхема, призначена для управління електронними пристроями (рис.2.6). Типовий мікроконтролер поєднує в собі функції процесора і периферійних пристроїв, може містити ОЗП і ПЗП. По суті, це однокристальний комп'ютер, здатний виконувати прості завдання. Опис виводів приведено в таблиці 2.1.



Риунок 2.6 - Схема контролера ATmega8

Таблиця 2.1.Опис виводів ATmega8

PB0	Вхід/вихід	цифровий порт PB0
ICP1	Вхід	захоплення вихід У 1
PB1	Вхід/вихід	цифровий порт PB1
OC1A	Вихід	вихід порівняння/ШИМ 1A
PB2	Вхід/вихід	цифровий порт PB2
OC1B	Вихід	вихід порівняння /ШИМ 1B

SS	Вхід	вхід Slave для SPI
PB3	Вхід/вихід	цифровий порт PB3
OC2	Выход	вихід порівняння /ШИМ 2
MOSI	Вхід/вихід	вхід даних в режимі Slave для SPI і ISP / вхід даних в режимі Master для SPI и ISP
PB4	Вхід/вихід	цифровий порт PB4
MISO	Вхід/вихід	вхід даних в режимі Master для SPI и ISP / вихід даних в режимі Slave для SPI и ISP
PB5	Вхід/вихід	цифровий порт PB5
SCK	Вхід/вихід	тактовий вхід в режимі Slave для SPI и ISP / тактовий вихід в режимі Master для SPI и ISP
PB6	Вхід/вихід	цифровий порт PB6 при роботі від 28монтованого генератора
XTAL1	Вхід	тактовий вхід, кварцовий або керамічний резонатор
TOSC1	Вхід	не використовується при роботі від зовнішнього генератора
PB7	Вхід/вихід	цифровий порт PB7 при роботі від вбудованого генератора
XTAL2	Вхід	для підключення кварцового або керамічного резонатора
TOSC2	Вхід	тактовий вихід при роботі від вбудованого генератора

Ядро Atmel AVR поєднує в собі багатий набір інструкцій з робочими 32 регістрів загального призначення. Всі 32 регістри безпосередньо підключені до арифметико Logic (ALU), дозволяючи дві незалежні регістри, які будуть доступні в одній інструкції, виконані в один такт. в результаті архітектура є більш ефективним код при досягненні пропускну здатність до десяти разів швидше в порівнянні з традиційними CISC мікроконтролерами. Atmega8 забезпечує наступні функції: 8 кбайт програмована флеш-пам’ять з Read-While-Write можливості, 512 байт EEPROM, 1 Кбайт SRAM, 23 загального призначення

ліній введення/виводу, 32 регістра загального призначення, працює три гнучких таймера / лічильники з порівняти. режими, внутрішні і зовнішні переривання, послідовний програмований USART, байт орієнтований двопровідний послідовний інтерфейс, 6-канальний АЦП (вісім каналів в TQFP і пакети QFN / МЗФ) з 10-бітна точність, програмований сторожовий таймер з вбудованим генератором, послідовний порт SPI, і п’ять режимів економії програмного забезпечення за вибором потужності. Режим очікування зупиняє CPU, дозволяючи статичне ОЗУ, таймер/лічильники, порт SPI і система переривань продовжують функціонувати. Powerdown Режим зберігає вміст регістрів, але заморожує осцилятор, відключаючи все інші функції чипу до наступного переривання або апаратного скидання. У режимі економії енергії, асинхронний таймер продовжує працювати, дозволяючи користувачеві підтримувати базу таймера в той час як інша частина пристрою спить. Режим придушення шумів АЦП зупиняє ЦПУ і всі модулі вводу/виводу, крім асинхронного таймера і АЦП, щоб звести до мінімуму перешкоди перемикання під час перетворення АЦП. У режимі очікування, кристал / резонатор генератор працює в той час як інша частина пристрою спить. Це дозволяє дуже швидкий пуск в поєднанні з низьким енергоспоживанням. Пристрій виготовляється з використанням незалежну технологію пам’яті високої щільності фірми Atmel.

Флеш-пам’ять Програма може бути перепрограмований In-System через послідовний інтерфейс SPI, шляхом направлення звичайна незалежна пам’ять програміста, або за допомогою програми завантаження на чіпі, що працюють на AVR ядро. Програма завантаження може використовувати будь-який інтерфейс для завантаження прикладної програми в Застосування флеш-пам’яті. Програмне забезпечення в Flash-секції завантаження буде продовжувати працювати в той час як Застосування спалаху Розділ оновлюється, забезпечуючи справжню операцію читання В той час запису. Об’єднавши 8-розрядний RISC-процесор з In-System Self-програмована флеш-пам’ять на монолітному чипі, то Atmel Atmega8 є потужним мікроконтролер, який забезпечує високо гнучке і економічне рішення для багатьох вбудованих додатків управління. Atmega8

підтримується з повним набором програмних і розвитку системи інструментів, в тому числі компілятори, макроассемблера, програмні імітатори і набори оцінки.

Порт B є порт 8-розрядний двонаправлений введення/виведення з внутрішніми навантажувальними резисторами (обраних для кожного біта). Вихідні буфери порту B мають симетричні характеристики приводу як з високою раковиною і джерелом можливість. В якості вихідних даних, порт B ніжка, які зовні насунутому буде джерелом струму, якщо підтягуючий резистори активуються. Ніжки порту B є три стани, коли умова скидання стає активним, навіть якщо годинник не працює. Залежно від параметрів вибору годин зупиняється. PB6 можуть бути використані в якості вхідних даних на інвертується Oscillator підсилювач та вхід в операційну ланцюг внутрішнього годинника. Залежно від параметрів вибору годин запобіжників, PB7 може бути використаний в якості висновку з інвертуючим. Осцилятори підсилювач. Якщо внутрішній калібрований RC генератор використовується в якості джерела тактових імпульсів, PB7..6 використовується як TOSC2..1 вхід для асинхронного таймера / Counter2 якщо біт AS2 в APCP встановлений.

Порт C є порт 7-розрядний двонаправлений введення/виведення з внутрішніми навантажувальних резисторів (обраних для кожного біта). Вихідні буфери порту C мають симетричні характеристики приводу як з високою раковиною і джерелом можливість. В якості вихідних даних, порт C ніжки, які зовні насунутому буде джерелом струму, якщо позитивні резистори активуються. Загальні ніжки порту C знаходяться в третьому стані, коли умова скидання стає активним, навіть якщо годинник не працює.

Порт D є порт 8-розрядний двонаправлений введення/виведення з внутрішніми навантажувальних резисторів (обраних для кожного біта). Вихідні буфери порту D мають симетричні характеристики приводу як з високою раковиною і джерелом можливість. В якості вихідних даних, порт D шпильки, які зовні насунутому буде джерелом струму, якщо негативні резистори активуються. Загальні ніжки порту D знаходяться в третьому стані, коли умова скидання стає активним, навіть якщо годинник не працює.

2.4.1 Годинник реального часу DS1307

Годинник реального часу з послідовним інтерфейсом DS1307 – це малоспоживаючий повний двійково-десятковий годинник-календар, що включає 56 байтів незалежної статичної ОЗУ. Адреси та дані передаються послідовно по двопровідній двобічній шині. Годинник-календар відраховують секунди, хвилини, години, день, дату, місяць і рік. Остання дата місяця автоматично коригується для місяців з кількістю днів менше 31, включаючи корекцію високосного року. Годинники працюють як в 24-годинному, так і в 12-годинному режимах з індикатором AM / PM. DS1307 має вбудовану схему спостереження за харчуванням, яка виявляє перебої харчування і автоматично перемикається на живлення від батареї. На рис. 12. Представлена блок-схема DS1307.

Характеристики

- Годинник реального часу (RTC) відраховують секунди, хвилини, години, дату місяця, місяць, день тижні і рік з компенсацією високосного року, дійсної до 2100 року;

- 56-байтовое незалежне ОЗУ з живленням від батареї для зберігання призначених для користувача даних;

- Двопровідний послідовний інтерфейс;

- Програмований вихідний сигнал з прямокутними імпульсами (для тактування зовнішніх пристроїв);

- Автоматичне виявлення падіння напруга і схема перемикання на батарею;

- Споживання менше 500 нА в режимі батарейній підтримки при працюючому тактовому генераторі;

- Промисловий діапазон температур: від – 40 ° С до + 85 ° С;

- Мікросхема проводиться в 8-вивідних корпусах DIP і SOIC.

Розташування висновків приведено на рис. 2.4.

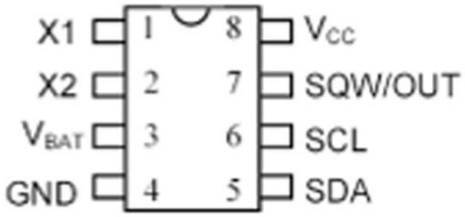


Рисунок 2.4 – DS1307 8-вивідних DIP

Опис виводів:

Vcc, GND – виводи для під'єднання до джерела живлення.

Vcc – це вхід +5 В.

Коли напруга живлення вище 1.25 * VBAT, пристрій повністю доступно, і можна виконувати читання і запис даних. Коли підключено батарея на 3 В, і Vcc нижче, ніж 1.25 * VBAT, читання і запис заборонені, проте функція відліку часу продовжує працювати. Як тільки Vcc падає нижче VBAT, ОЗУ і RTC переключаються на батарейне харчування VBAT.

VBAT – вхід для будь-якої стандартної тривольтової літієвої батареї або іншого джерела енергії. Для нормальної роботи DS1307 необхідно, щоб напруга батареї було в діапазоні 2.0 … 3.5 В. Літієва батарея з ємністю 48 мА / год або більше при відсутності харчування буде підтримувати DS1307 в. Протягом більше 10 років при температурі 25 ° С.

SCL (Serial Clock Input – вхід послідовних синхроімпульсів) – використовується для синхронізації даних по послідовному інтерфейсу.

SDA (Serial Data Input / Output – вхід / вихід послідовних даних) – висновок входу / виходу для двухпроводного послідовного інтерфейсу. Висновок SDA – з відкритим стоком і вимагає зовнішнього підтягує резистора. SQW / OUT (Square Wave / Output Driver – сигнал з прямокутними імпульсами) – коли включений, тобто біт SQWE встановлений в 1, висновок SQW / OUT видає прямокутні імпульси з однієї з чотирьох DS1307 Технічний опис частот (1 Гц, 4 кГц, 8 кГц, 32 кГц). Вивід SQW / OUT – з відкритим стоком і вимагає зовнішнього підтягує резистора. SQW / OUT буде працювати як при харчуванні від Vcc, так і при харчуванні від VBAT.

X1, X2 – виводи для стандартного кварцового резонатора з частотою 32.768 кГц. Схема внутрішнього тактового генератора розроблена для роботи з кварцовим резонатором, що має вхідну ємність 12.5 пФ.

RTC та карта адресації пам’яті. Карта адрес для RTC і регістрів ОЗП представлена на рисунку. 13.

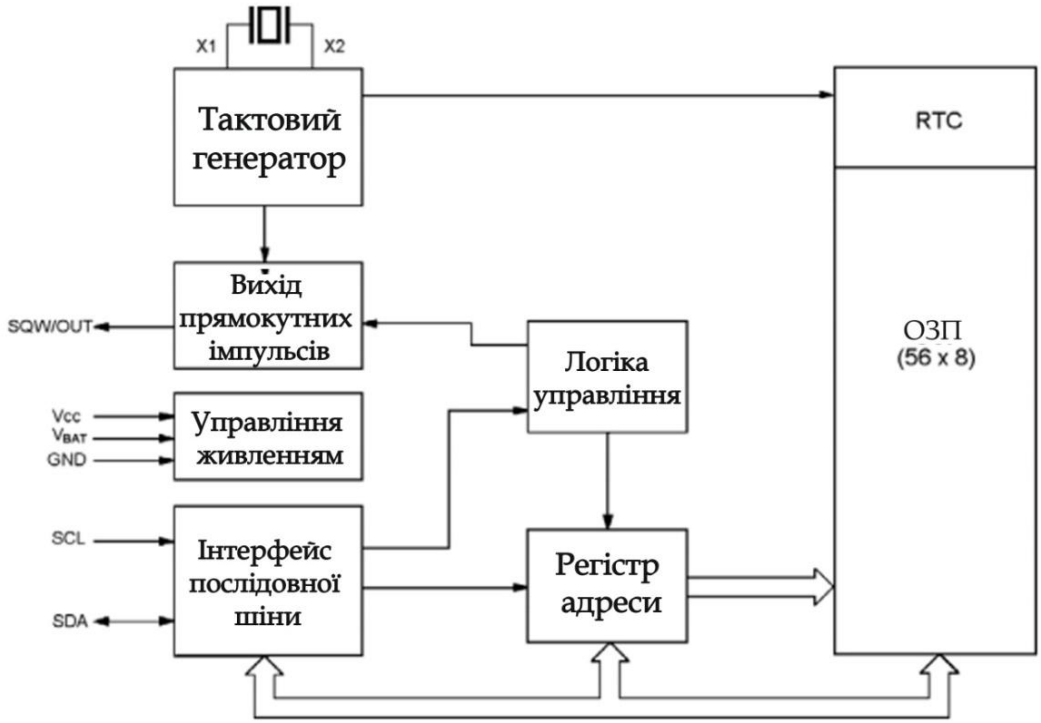


Рисунок 2.5 – Блок-схема DS1307

Регістри RTC розташовані в осередках адрес від 00h до 07h. Регістри ОЗП розташовані в осередках адрес від 08h до 3Fh. В процесі багатобайтового доступу, коли адресний покажчик досягає 3Fh (кінець простору ОЗП), він переміщується на осередок 00h – початок простору RTC.

Рисунок 2.6 – Карта адрес DS1307

2.5 Архітектура програмної частини

Програмна частина складається з двох режимів:

1) Робочий режим приведений на рис. 2.3.
2) Режим імітування приведений на рис 2.4.

Робочий режим, при якому програма отримавши дані з апаратної частини одразу будує точковий графік, зберігає отримані дані в окремий txt файл, видає користувачу реальний час на ПК, може вибрати середню швидкість вітру у трьох варіаціях (середня швидкість за одну годину, за 3 години та 6 годин), також видає користувачу поточний час і швидкість отриману з контролеру та можливість корегування часу годинника реального часу апаратної частини. На рисунок 2.7. приведена структурна схема робочого режиму. На рисунок 2.8. приведена структурна схема робочого режиму.

Режим тестування, при якому програма отримує дані не з апаратної частини, а з інтерфейсу користувача. А саме замість поточного часу і швидкості вітру отриманих з апаратної частини, програма бере час з годинника реального часу ПК, а швидкість руху вітру з поточного положення повзунка. Нульове значення повзунка відповідає за нульове значення вітру, а максимальне значення повзунка дорівнює максимальному значенню вітру. При цьому програма будує

точковий графік як і в робочому режимі, зберігає дані в окремому txt файлі та користувач може вибрати середню швидкість вітру у трьох варіаціях (середня швидкість за одну годину, за 3 години та 6 годин). Інтерфейс програмної частини приведено на рисунку 2.9.

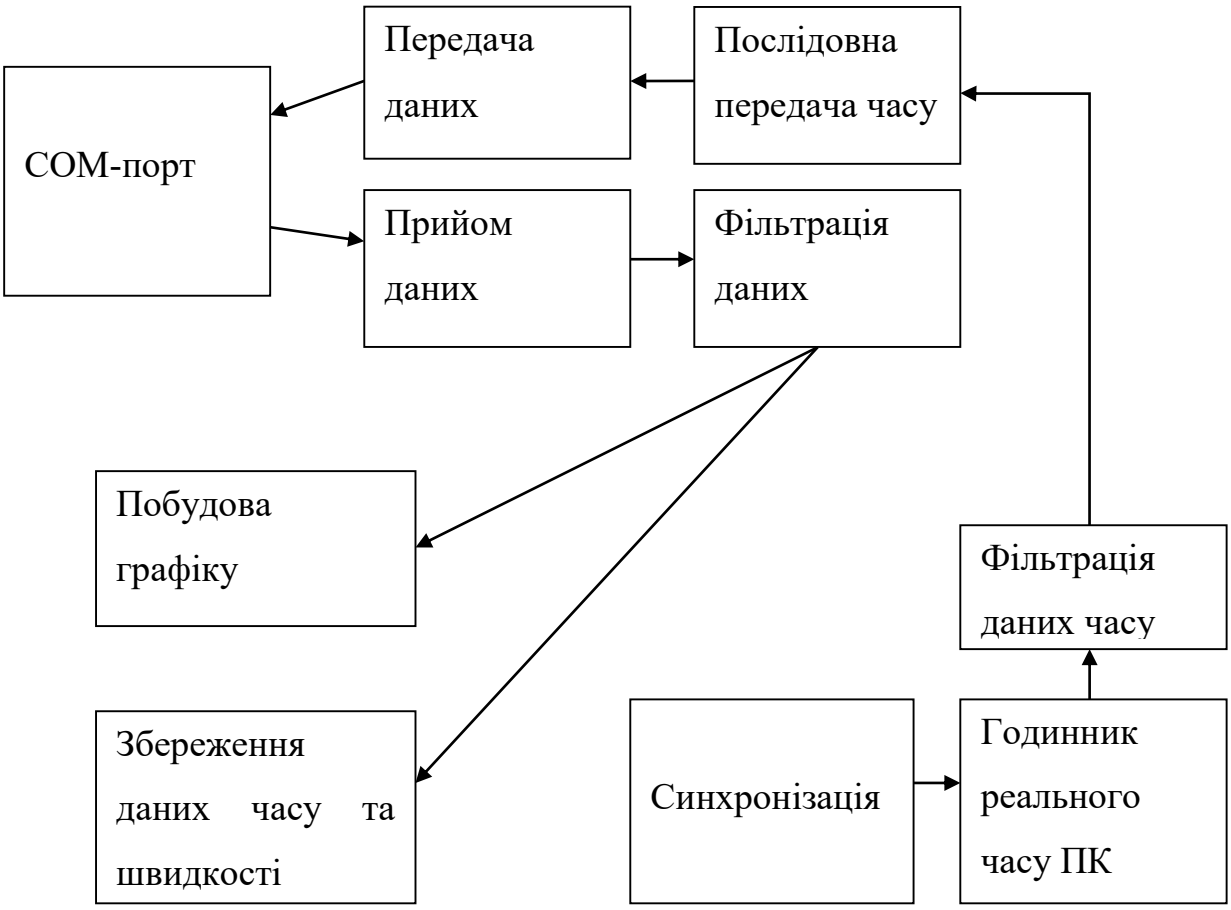


Рисунок 2.7 – Структурна схема робочого режиму

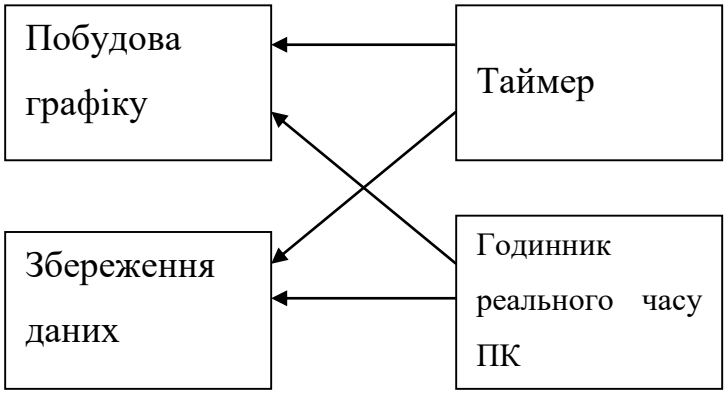
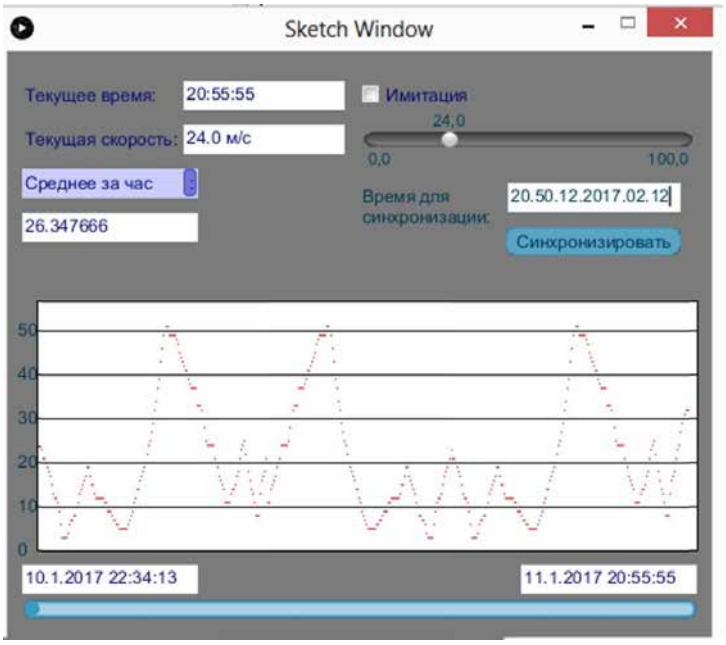
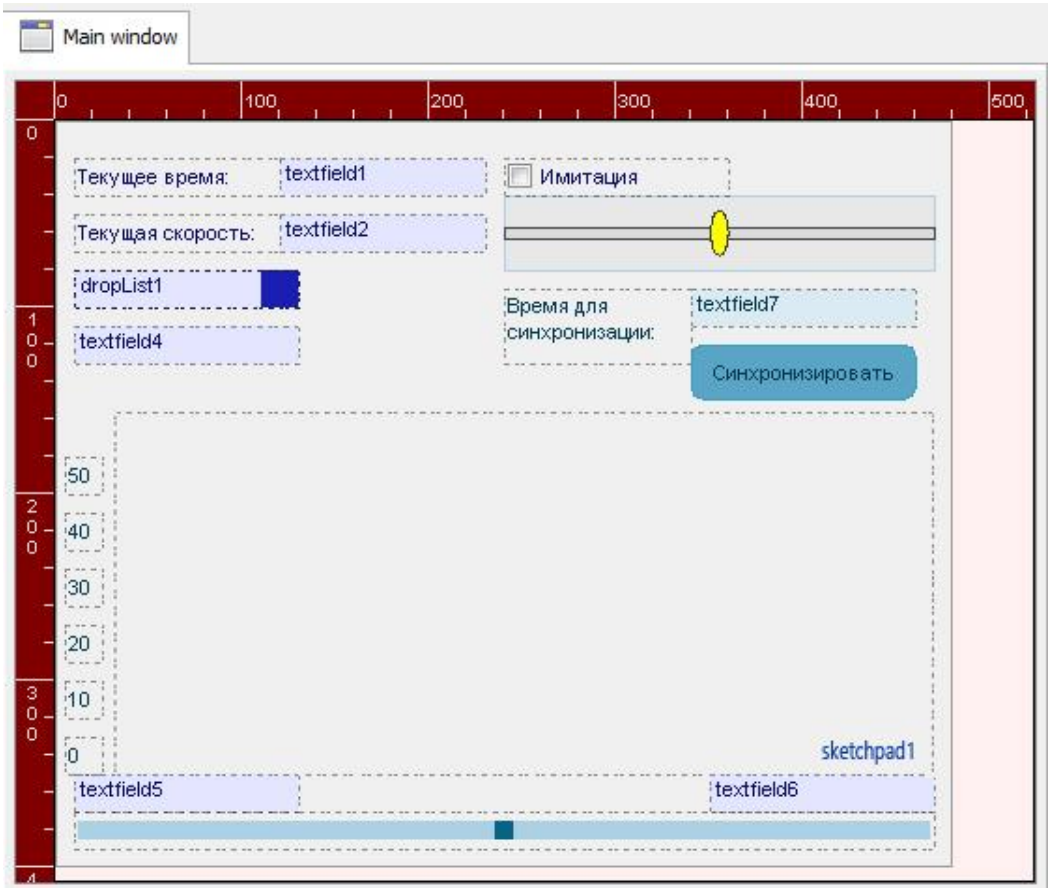


Рисунок 2.8 – Структурна схема програмної частини в режимі імітація



Рисуноку 2.9 – Інтерфейс програмної частини



Таблиця 2.2 – Опис функціональних елементів

Елемент	Значення
Label1	Назва – Поточний час
Labe2	Назва – Поточна швидкість
Labe3	Назва – Час для синхронізації
Label4	Назва – 0
Label5	Назва – 10
Labe6	Назва – 20
Labe7	Назва – 30
Labe8	Назва – 40
Labe9	Назва – 50
ChekBox1	Елемент прапорець
TextFild1	Текстове поле - Поточний час
TextFild2	Текстове поле - Поточна швидкість

TextFild3	Текстове поле – Середня швидкість за проміжок часу
TextFild4	Текстове поле – Значення часу з апаратної частини
TextFild5	Текстове поле – Значення поточного часу ПК
Button1	Кнопка для синхронізації часу для апаратної частини
DropList1	Випадаючий список для вибору часу за проміжок часу
SketChpad1	Точкового графік
Costom_slider1	Рухомий повзунок для корегування значення швидкості вітру

3 РОЗДІЛ РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір мови програмування для апаратної частини

При виборі мови програмування для апаратної частини були дотримані наступні критерії: швидкодія обрахунку даних, стабільність роботи, малий розмір програми, велика ефективність. За цими критеріями підійшла мова програмування Сі.

Для мікроконтролерів існують різні мови програмування, але, найбільш придатними є Асемблер і Сі, оскільки в цих мовах в найкращій мірі реалізовані всі необхідні можливості по управлінню апаратними засобами мікроконтролерів. Асемблер програє Сі в швидкості і зручності розробки програм, але має помітні переваги в розмірі кінцевого виконуваного коду, а відповідно, і швидкості його виконання.

Сі - це універсальна мова програмування з компактним варіантом запису виразів, сучасними механізмами управління структурами даних і багатим набором операторів.

Основні переваги Сі перед ассемблером: висока швидкість розробки програм; універсальність, що не вимагає досконального вивчення архітектури мікроконтролера; найкраща документовані і читаність алгоритму; наявність бібліотек функцій; підтримка обчислень з плаваючою точкою.

На допомогу у створенні програми апаратної частин є Code Vision AVR C Compiler. Компілятор поставляється разом з інтегрованим середовищем розробки, в яку, крім стандартних можливостей, включена досить цікава функція - CodeWizardAVR Automatic Program Generator. Наявність в середовищі розробки послідовного терміналу дозволяє виробляти налагодження програм з використанням послідовного порту мікроконтролера.

Слід ще раз відзначити, що архітектура і система команд AVR створювалася за безпосередньої участі розробників компілятора мови Сі і в ній враховані особливості цієї мови. Компіляція вихідних текстів, написаних на Сі, здійснюється швидко і дає компактний, ефективний код.

3.2 Реалізація програмної частини

Створення програми апаратної частини починається з визначення структурної складової і принципової схеми та рішення як повинен взаємодіяти контролер з комп'ютером. Визначення та реалізація математичного розрахунку перекладів в потрібні одиниці вимірювання, перетворення бінарного коду в десятинний, функцій передачі даних від мікроконтролера до комп'ютера.

Для полегшення написання програми створені макроси, для простоти взаємодії користувача з кодом програми. А саме макроси Tarirovka, sek_1, i2cwrikorr, i2cwrit, i2cred.

При написанні програми визначаємося з бібліотеками які нам допоможуть в реалізації та полегшення написання коду програми:

<stdio.h> визначає початкові налаштування програми під мікроконтролер;

<math.h> реалізує роботи програми з математичними функціями;

<avr/delay.h> робота із затримками;

<avr\interrupt.h> підтримує апаратні переривання;

<avr\lcd.h> робота з lcd дисплеєм;

Задаємо змінні Volatile для зберігання бінарних даних:

(ss) секунди;

(мм) хвилини;

(НН) години;

Змінні для зберігання даних після перетворення бінарних чисел часу в десяткові одиниці ss_des, mm_des, HH_des, ss_ed, mm_ed, HH_ed. Ці змінні також приймають участь відображенні на LCD часу.

Змінні для обробки вимірювання швидкості вітру: IMPULSU підрахунок імпульсів датчика, IMPULSU_DATCHIK змінна для математичних розрахунків, при обчислюванні швидкості вітру та коефіцієнту ділення. Змінна COEFICIENT зберігає коефіцієнт розрахунку швидкості вітру. Змінна OBRAZCOVKA для читання коефіцієнту з EPROM. Також змінна VTER зберігає значення вітру у Float (значення з плаваючою комою). SKOROST_VTERA строкова змінна значення Float. IMP_SEK зміна перерахунку секундних імпульсів. AAAA, MMM, BBBB, HHHH строкові змінні даних для передачі через UART \ RS-232.

Наступним кроком налаштування регістрів та портів мікроконтролеру на приймання \ передачу даних:

```
{
DDRC=0b11111111;
DDRD=0b11001101;
DDRB=0b11111110;
PORTB=0b00000001;
PORTC=0b00000000;
PORTD=0b11101101;
}
```

Наступним кроком є налаштування первинного годинника через порт I2C на роботу з передачою ними байт часу. Після цих маніпуляцій запускається мікропрограма прийому даних з первинного годинника і збереження їх в змінних. При кожному перериванні порту імпульсами датчика програма заходить в макрос, який підраховує імпульси для подальшої їх обробці. При перериванні секундних імпульсів видаються первинними годинниками відбувається перехід програми в макрос перерахунку імпульсів в швидкість вітру.

```
    void FCM_sek_1()
   {
  FCV_SEKI = FCV_SEKI + 1;
  FCV_IMPULSU = FCV_IMPULS_DATCHIK;
  FCV_IMPULS_DATCHIK = 0;
  FCV_SKOROST_VETRA = flt_fromi(FCV_IMPULSU
/FCV_KOEFICIENT);
   }
```

При опитуванні кнопки тарировка програма заходить в макрос Tarirovka. В цій функції відбувається вимір і підрахунок імпульсів за п'ять секунд, після чого знаходить середнє значення за одну секунду (призначене для малої швидкості вітру). Мала швидкість вітру призначена для значення датчика, усереднюється через неточності механічної частини. В цей же час програма висвічує транспарант тарування з підрахунком вхідних імпульсів:

```
FCD_LCDDisplay0_PrintString("TARIRIVKA", 9);
```

Після закінчення п'яти секунд цикл підрахунку закінчується і пропонує вибрати кнопками швидкість вітру, при якій проводилися вимірювання:

```
 while (FCV_SEKI < 5) ;
```

Якщо ніяких дій не робилося або кнопка відпущена більше 4 секунд програма заходить в підрахунок коефіцієнта і при цьому висвічується транспарант – Koficient =, і це значення записується в виділену комірку пам'яті Eprom:

```
FCV_KOEFICIENT = FCV_IMP_ZA_SEK / 5 / FCV_OBRAZCOVKA;
FCD_LCDDisplay0_PrintString("KOEFFICIENT", 11);
```

Основний цикл програми полягає в опитуванні кнопок: Корекція хвилин, годин (кнопки хв., год., при тарировці мають функцію виставлення швидкості вітру). А також переривання перериваються імпульсами датчика та секундними імпульсами первинних годин (кожне переривання працює зі своїм макросом).

В основному циклі йде перетворення часу довічного в десяткове і виведенням його на LCD-дисплей. Передача даних по Uart \ RS-232. Проводитися тільки після перетворення змінних з integer і float в змінну з розширенням string.

```
FCI_FLOAT_TO_STRING(FCV_SKOROST_VETRA, 6,
FCV_VTER,FCSZ_VTER);
FCD_RS2320_SendRS232String(FCV_VTER, FCSZ_VTER);
FCD_RS2320_SendRS232Char(0x0D);
```

3.3 Вибір мови програмування для програмної частини

При виборі мови програмування програмної частини дотримані наступні критерії - незалежність від платформи на якій буде встановлена програма, стабільна робота. Вибір впав на мову програмування JavaScript. Одна з головних переваг мови JavaScript - код можна запускати під управлінням операційних систем Windows, Linux, FreeBSD, Solaris, Apple Mac і ін. Це є важливим фактором вибору мови програмування, коли програми завантажуються за допомогою глобальної мережі Інтернет і використовуються на різних платформах.

На допомогу в створенні програмної частини на мові програмування JavaScript приходить програма Processing. Processing - розширена мова програмування, заснована на мові програмування JavaScript, вона являє собою легкий і швидкий інструментарій для людей, які хочуть програмувати зображення, анімацію і інтерфейси.

Processing також дає можливість швидко і легко створювати мультимедіа програми (в термінології processing - скетчі). Скетч - вихідний файл вашої програми, який дозволяє розробляти графіки, анімацію, різноманітну візуалізацію та інтерактивні додатки.

Користувачу нічого не заважає створювати навіть 3D-аплікації (в тому числі і ігри), адже processing має засоби підтримки OpenGL. Всі ці можливості,

укупі з великою кількістю функцій і дуже логічним синтаксисом, роблять цю мову ідеальним для навчання та прищеплення інтересу до програмування.

3.4 Реалізація програмної частини

При створенні програми для програмної частини треба, перш за все починається з визначення структурної складової і принципової схеми та рішення як повинен взаємодіяти комп'ютер з контролером. Визначення та реалізація математичного розрахунку перекладів в потрібні одиниці вимірювання, функцій передачі даних від комп'ютера до мікроконтролера, побудова точкового графіку, збереження отриманого значення в окремий txt файл, видавати користувачу реальний час на ПК, також можливість вибрати середню швидкість вітру у трьох варіаціях (середня швидкість за одну годину, за 3 години та 6 годин). Також видає користувачу поточний час ПК і швидкість отриману з контролеру, можливість корегування часу годинника реального часу апаратної частини.

Перш за все створюємо точка входу в компілятор, який дозволяє користувачу працювати за графічним інтерфейсом, отримувати дані з COM-портом, переклад значення в текстові поля поточного часу TextFild1 та поточної швидкості вітру TextFild2. Значення з контролера поточного часу об'єднуються в строкове значення через роздільники ":", а значення поточної швидкості одразу переводиться в значення string та додається до нього " м/с":

```
public void textfield1_change1(GTextField source, GEvent event) {
 textfield1.setText(hour() + ":" + minute() + ":" + second());
}
public void textfield2_change1(GTextField source, GEvent event) {
textfield2.setText(str(speed) + " м/с");
}
```

Як раніше описувались в вимогах до інтерфейсу що нам потрібно обробляти дані отримані з апаратної частини, через ChekBox1 і обробляти отримані дані та

видавати користувачу значення за годину, за 3 години, та 6 годин. Описуємо випадаючий список через представляння узагальненої корекції, яка успадковує свою функціональність від класу AbstractList і застосовує інтерфейс List. Який представляє собою простий список, аналогічний масиву, за тим винятком, що кількість елементів в ньому не фіксоване.

```
  public void dropList1_click1(GDropList source, GEvent event) {
  if (dropList1.getSelectedIndex() == 0){
  }
  if (dropList1.getSelectedIndex() == 1){
  //redrawRate = 300;
  historySize = 60 * 24;
  }
  if (dropList1.getSelectedIndex() == 2){
  historySize = 60 * 24 * 31;
  //redrawRate = 600;
 }
```

Побудова графіку є невід'ємною частиною як для робочого режиму так і для режиму імітації. Так, що розглянемо елемент checkbox1- прапорець, генерує дві події, при одній checkbox1 переходить в режим імітації, запускаючи годинник реального часу ПК, та активізує Costom_slider1. Також кожна секунда є тактом і додає до змінної speedHistory.size() - xSize + 1, значення початкової точки часу в режимі імітації.

```
 if (speedHistory.size()>xSize) {
  slider1.setLimits(0, speedHistory.size() - xSize);
  offset = speedHistory.size() - xSize + 1;
 } else {
  slider1.setLimits(0, 1);
  offset = 0;
```

```
  }
```

Значення графіку при отриманні даних в режимі імітації швидкість перемінних x та y

```
    if (offset + index < speedHistory.size()) {
```

Збереження змінних wind та speed після функції speedJ.setFloat

зберігає значення в числовому десятковому вигляді, і файл зберігається з назвою

```
void dispose() {
  saveJSONArray(speedHistory, "wind_speed_history.json");
}
```

Знач

```
      textfield5.setText(speedHistory.getJSONObject(offset).getString("time"));
```

4 ВИСНОВКИ

В ході виконання кваліфікаційної роботи, було розглянуто існуючі програмно-апаратні засоби виміру вітру, на прикладі WeatherLink USB 6510, портативного приладу TFA426002 Vaavud, анемометру AZ8919, метеостанції Oregon Scientific BAR806. Досліджено переваги та недоліки кожного з них. Проаналізувавши їх, було виявлено, що ці прилади не повністю задовольняють потреби користувачів. Суттєвими недоліками є відсутність підключення до ПК та Інтернету, велика вартість, громісткість, складність у використанні інтерфейсу.

На основі цих досліджень, було прийнято рішення розробити програмно-апаратний засіб на базі мікроконтролера ATmega8 та годинника реального часу DS1307, де враховані всі недоліки вище згаданих приладів. В ході створення програмно-апаратного засобу виміру вітру були створені структурні і принципові схеми; описані алгоритми обробки даних на ПК та LCD дисплею, передача даних на COM-порт, корегування годинника реального часу через програмний інтерфейс користувачем та зберігання даних часу і швидкості вітру в окремому файлі.

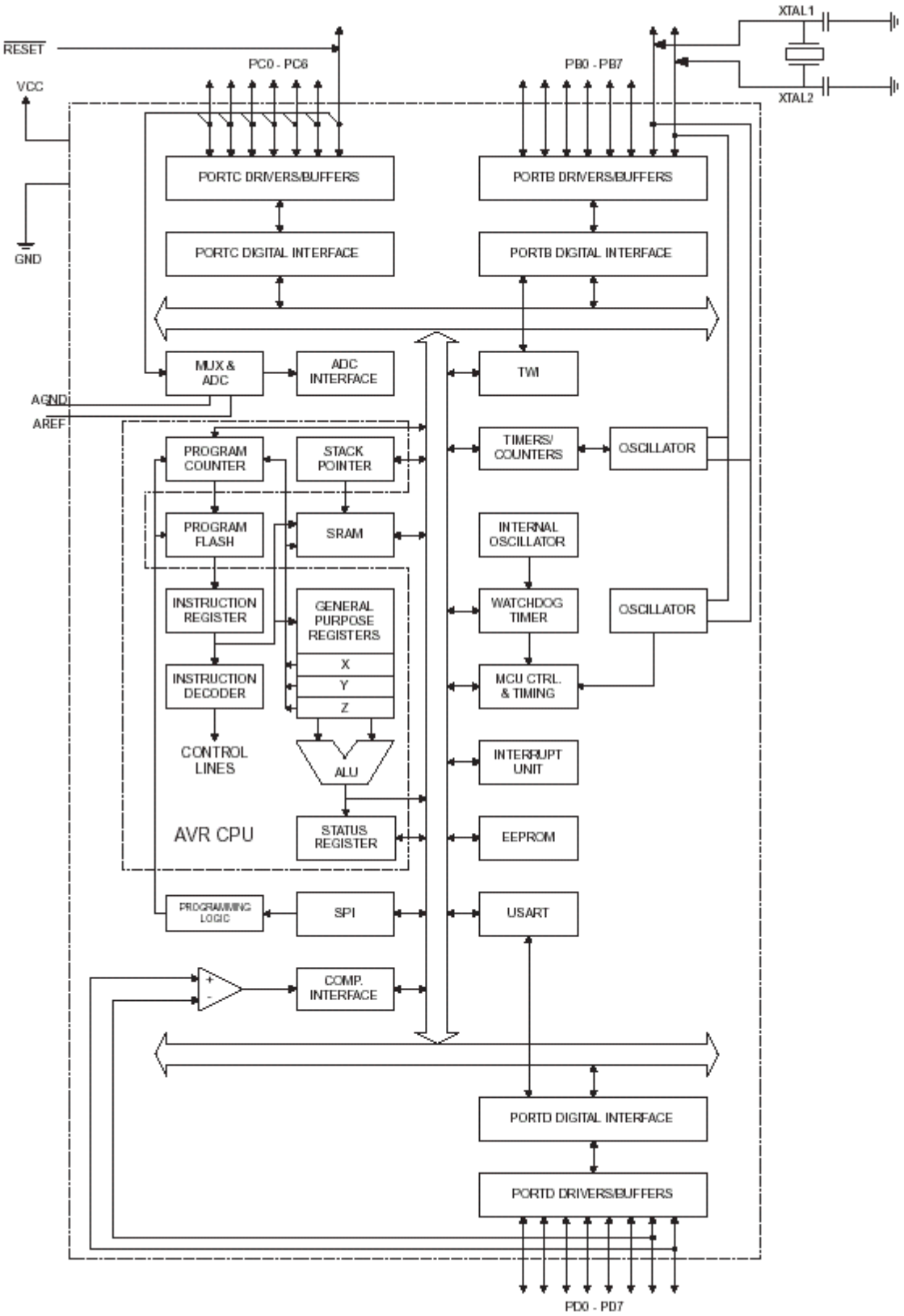
Було розглянуто дві мови програмування C та JavaScript. Мова C обрана для створення апаратної частини. За допомогою неї було реалізовано взаємодію контролера, годинника реального часу з LCD дисплем та обробку, прийом/передачу даних на ПК. В свою чергу JavaScript використовувався для створення програмної частини. Вона взаємодіє з користувачем через інтерфейс, який дає можливість обробляти дані з апаратної частини для побудови графіка, зберігання отриманих результатів на ПК та корегування годинника реального часу апаратної частини.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Davis Instruments та інтерфейс для персонального комп'ютера WeatherLink USB 6510 [Електронний ресурс]. URL: http://davis.kiev.ua/scope/weatherlink
2. Метеостанції [Електронний ресурс]. URL: http://meteostantsiya.com.ua/gadzhety/tfa-426002-vaavud-detail
3. Міхєєв А.В., Демченко С.О. Мікроконтролери вбудованих систем: Arduino та його застосування. – Харків: ХНУРЕ, 2020. – 152 с.
4. Струк А.О., Яворський І.П. Основи цифрової електроніки та мікропроцесорної техніки: навч. посіб. – Львів: Видавництво ЛНУ ім. І. Франка, 2021. – 264 с.
5. Maksimovic D., Erickson R. Fundamentals of Power Electronics. – Springer, 2020. – 912 p.
6. Симонович С.В. Основи мікропроцесорної техніки. – Х.: Основа, 2018. – 296 с.
7. Мережа магазинів радіоелектроніки RC.ua: технічні специфікації модулів [Електронний ресурс]. – Режим доступу: https://rc.ua

Додаток А

Блоксхема контролера Atmega 8



Додаток Б

Код апаратної частини

```
//аппаратна частина
//бібліотеки
#define F_CPU 8000000UL
#include <stdio.h>
#include <math.h>
#include <avr/delay.h>
#include <avr\interrupt.h>
#include <avr\lcd.h>
// змінні
volatile MX_UINT8 FCV_SEKSEK;
volatile MX_UINT8 FCV_SS;
volatile MX_UINT8 FCV_MM;
volatile MX_UINT8 FCV_HH;
volatile MX_UINT8 FCV_AAAA;
volatile MX_UINT8 FCV_SEKI = (0x0);
volatile MX_UINT8 FCV_G;
volatile MX_UINT8 FCV_OBRAZCOVKA = (0x0);
volatile MX_UINT8 FCV_HHH = (0x0);
volatile MX_UINT8 FCV_SS_DES;
volatile MX_UINT8 FCV_KOEFICIENT;
volatile MX_SINT16 FCV_MM_DES;
volatile MX_SINT16 FCV_HH_DES;
volatile MX_BOOL FCV_Z = (0);
volatile MX_BOOL FCV_KN_KORREKCII2;
volatile MX_CHAR FCV_M1[FCSZ_M1];
volatile MX_CHAR FCV_M2[FCSZ_M2];
volatile MX_UINT32 FCV_IMP_ZA_SEK = (0x0);
volatile MX_UINT8 FCV_KNN;
volatile MX_UINT8 FCV_MMM = (0x0);
volatile MX_UINT8 FCV_SS_ED;
volatile MX_CHAR FCV_M[FCSZ_M];
volatile MX_SINT16 FCV_MM_ED;
volatile MX_SINT16 FCV_HH_ED;
volatile MX_BOOL FCV_KOR;
volatile MX_CHAR FCV_VTER[FCSZ_VTER];
volatile MX_CHAR FCV_H1[FCSZ_H1];
volatile MX_CHAR FCV_H2[FCSZ_H2];
volatile MX_UINT16 FCV_IMPULS_DATCHIK = (0x0);
```

```
volatile MX_UINT16 FCV_IMPULSU = (0x0);
volatile MX_FLOAT FCV_SKOROST_VETRA = (0.0);
volatile MX_BOOL FCV_IMP_SEK = (0);
volatile MX_BOOL FCV_KN_KORREKCII;
volatile MX_UINT8 FCV_BBBBB;
// Назначення виводів
int main(void)
{
DDRC=0b11111111;
DDRD=0b11001101;
DDRB=0b11111110;
PORTB=0b00000001;
PORTC=0b00000000;
PORTD=0b11101101;
}
      //Макрос коррегування часу поі2с
      void FCM_i2cwrit()
    {
                  FCD_I2C_Master0_MI2C_Init();
                  FCD_I2C_Master0_MI2C_Start();
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0xD0);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(FCV_MMM);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(FCV_HHH);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0x10);
                  FCD_I2C_Master0_MI2C_Transmit_Byte(0B10010000);
                  FCD_I2C_Master0_MI2C_Stop();
    }
        //Макрос читання часу поI2C
  void FCM_i2cred()
          {
                        FCD_I2C_Master0_MI2C_Start();
                        FCD_I2C_Master0_MI2C_Transmit_Byte(0xD0);
                        FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
                        FCD_I2C_Master0_MI2C_Restart()
                        FCD_I2C_Master0_MI2C_Transmit_Byte(0xD1);
                        FCV_SS = FCD_I2C_Master0_MI2C_Receive_Byte(0);
```

```
                    FCV_MM = FCD_I2C_Master0_MI2C_Receive_Byte(0);
                    FCV_HH = FCD_I2C_Master0_MI2C_Receive_Byte(1);
                    FCD_I2C_Master0_MI2C_Stop();
    }
        //макрос посчет импульсов датчика
     void FCM_skorost()
      {
                        FCV_IMPULS_DATCHIK = FCV_IMPULS_DATCHIK + 1;
                        FCV_IMP_SEK = 1;
    }
       //макрос підрахування часу
         void FCM_sek_1()
   {
    FCV_SEKI = FCV_SEKI + 1;
    FCV_IMPULSU = FCV_IMPULS_DATCHIK;
    FCV_IMPULS_DATCHIK = 0;
    FCV_SKOROST_VETRA = flt_fromi(FCV_IMPULSU / FCV_KOEFICIENT);
   }
     //макрос тарировка
      void FCM_tarirovka()
            {
           FCV_AAAA = 0;

           while (FCV_SEKI < 5)
                    }
            if (FCV_IMP_SEK == 1)
            {
                    FCV_IMP_ZA_SEK = FCV_IMP_ZA_SEK + 1;
                    FCV_IMPULS_DATCHIK = 0;
                    FCV_IMP_SEK = 0;
           }
  //очищаем экран,  выставляем курсор, выводим данные
            FCD_LCDDisplay0_Clear();
            FCD_LCDDisplay0_Cursor(0, 1);
            FCD_LCDDisplay0_PrintString("TARIRIVKA", 9);
            FCD_LCDDisplay0_PrintNumber(FCV_IMP_ZA_SEK);
            delay_ms(200);
    }
    //  impuls_datchik = 0
    FCV_IMPULS_DATCHIK = 0;
//очищаем экран,  выставляем курсор, выводим данные
    FCD_LCDDisplay0_Clear();
```

```
FCD_LCDDisplay0_Cursor(0, 1);
FCD_LCDDisplay0_PrintString("TARIRIVKA", 9);
FCD_LCDDisplay0_PrintNumber(FCV_IMP_ZA_SEK);
delay_s(3);
while (FCV_AAAA < 10)
{
        //  aaaa = aaaa + 1
        FCV_AAAA = FCV_AAAA + 1;
        // kn_korrekcii
        FCV_KN_KORREKCII = FCD_SWITCH0_ReadState();
        delay_ms(10);
        // kn_korrekcii = 1?
        if (FCV_KN_KORREKCII == 1)
        {
                //  OBRAZCOVKA = OBRAZCOVKA + 1
                FCV_OBRAZCOVKA = FCV_OBRAZCOVKA + 1;
                FCV_AAAA = 0;
        } else {
                FCV_KN_KORREKCII2 = FCD_SWITCH1_ReadState();
                delay_ms(10);
                // kn_korrekcii2 = 1?
                if (FCV_KN_KORREKCII2 == 1)
                {
                        //  OBRAZCOVKA = OBRAZCOVKA - 1
                        //  aaaa = 0
                        FCV_OBRAZCOVKA = FCV_OBRAZCOVKA - 1;
                        FCV_AAAA = 0;
                }
        }
        FCD_LCDDisplay0_Clear();
        FCD_LCDDisplay0_Cursor(0, 0);
        //String("SPEED")
        FCD_LCDDisplay0_PrintString("SPEED", 5);
        FCD_LCDDisplay0_PrintNumber(FCV_OBRAZCOVKA);
        delay_ms(100);
}
//  koeficient = imp_za_sek / 5 / OBRAZCOVKA
FCV_KOEFICIENT = FCV_IMP_ZA_SEK / 5 / FCV_OBRAZCOVKA;
FCD_LCDDisplay0_Clear();
FCD_LCDDisplay0_Cursor(0, 1);
FCD_LCDDisplay0_PrintString("KOEFFICIENT", 11);
FCD_LCDDisplay0_PrintNumber(FCV_KOEFICIENT);
```

```
        delay_s(2);
        //  imp_za_sek = 0
        //  Сек = 0
        //  aaaa = 0
                FCV_IMP_ZA_SEK = 0;
                FCV_SEKI = 0;
                FCV_AAAA = 0;
                }
        void FCM_i2swritkorr()
{
        FCD_I2C_Master0_MI2C_Init();
        FCD_I2C_Master0_MI2C_Start();
        FCD_I2C_Master0_MI2C_Transmit_Byte(0xD0);
        FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
        FCD_I2C_Master0_MI2C_Transmit_Byte(0x00);
        FCD_I2C_Master0_MI2C_Transmit_Byte(FCV_MMM);
        FCD_I2C_Master0_MI2C_Transmit_Byte(FCV_HHH);
        FCD_I2C_Master0_MI2C_Transmit_Byte(1);
        FCD_I2C_Master0_MI2C_Stop();
}
int main()
{
        //Initialization
        MCUCSR=0x00;
        wdt_disable();
                RS232_8_UART_Init( );           //Виклик initialise function
        //Interrupt initialization code
        MCUCR |= (1 << ISC11) | (1 << ISC10);
        sei();
        GICR |= (1 << INT1);
        MCUCR |= (1 << ISC01) | (1 << ISC00);
        sei();
        GICR |= (1 << INT0);
        FCD_LCDDisplay0_Start();
        FCD_LCDDisplay0_Clear();
        while (1)
        {
                FCV_KN_KORREKCII = FCD_SWITCH0_ReadState();
                if (FCV_KN_KORREKCII == 1)
                {
                        FCV_MMM = FCV_MMM + 1;
                        FCM_i2swritkorr();
```

```
                    }
                    FCV_KN_KORREKCII = FCD_SWITCH1_ReadState();
                    if (FCV_KN_KORREKCII == 1)
                    {
                FCV_HHH = FCV_HHH + 1;
                FCM_i2swritkorr();
            }
            FCV_KOR = FCD_SWITCH2_ReadState();
            if (FCV_KOR == 1)
                {
                FCM_tarirovka();
                }
            FCM_i2cred();
            //  hh_des = hh >> 0x04
            //  hh_ed = hh & 0x0f
            //  mm_des = mm >> 0x04
            //  mm_ed = mm & 0x0f
            //  ss_des = ss >> 0x04
            //  ss_ed = ss & 0x0f
            FCV_HH_DES = FCV_HH >> 0x04;
            FCV_HH_ED = FCV_HH & 0x0f;
            FCV_MM_DES = FCV_MM >> 0x04;
            FCV_MM_ED = FCV_MM & 0x0f;
            FCV_SS_DES = FCV_SS >> 0x04;
            FCV_SS_ED = FCV_SS & 0x0f;
            FCD_LCDDisplay0_Clear();
            FCD_LCDDisplay0_Cursor(0, 0);
            FCD_LCDDisplay0_PrintNumber(FCV_HH_DES);
            FCD_LCDDisplay0_PrintNumber(FCV_HH_ED);
            FCD_LCDDisplay0_PrintString("", 0);
            FCD_LCDDisplay0_Cursor(3, 0);
            FCD_LCDDisplay0_PrintNumber(FCV_MM_DES);
            FCD_LCDDisplay0_PrintNumber(FCV_MM_ED);
            FCD_LCDDisplay0_PrintString("M", 1);
            FCD_LCDDisplay0_PrintNumber(FCV_SS_DES);
            FCD_LCDDisplay0_PrintNumber(FCV_SS_ED);
            FCD_LCDDisplay0_Cursor(0, 1);
            FCD_LCDDisplay0_PrintString("M/S", 3);
FCD_LCDDisplay0_PrintNumber(flt_toi(FCV_SKOROST_VETRA));
            delay_ms(100);
            FCI_FLOAT_TO_STRING(FCV_SKOROST_VETRA, 6, FCV_VTER,FCSZ_VTER);
            FCD_RS2320_SendRS232String(FCV_VTER, FCSZ_VTER);
            FCD_RS2320_SendRS232Char(0x0D);
```

```
                delay_ms(255);
                delay_ms(45);
        }
}
```

Додаток В

Код програми програмної частини

```
public void textfield1_change1(GTextField source, GEvent event) { //_CODE_:textfield1:236497:
  //println("textfield1 - GTextField >> GEvent." + event + " @ " + millis());
  textfield1.setText(hour() + ":" + minute() + ":" + second());
} //_CODE_:textfield1:236497:
public void textfield2_change1(GTextField source, GEvent event) { //_CODE_:textfield2:342463:
  //println("textfield2 - GTextField >> GEvent." + event + " @ " + millis());
  textfield2.setText(str(speed) + " м/с");
} //_CODE_:textfield2:342463:

public void custom_slider1_change1(GCustomSlider source, GEvent event) { //_CODE_:custom_slider1:694590:
  //println("custom_slider1 - GCustomSlider >> GEvent." + event + " @ " + millis());
} //_CODE_:custom_slider1:694590:

public void checkbox1_clicked1(GCheckbox source, GEvent event) { //_CODE_:checkbox1:700584:
  //println("checkbox1 - GCheckbox >> GEvent." + event + " @ " + millis());
  startDraw = !startDraw;
} //_CODE_:checkbox1:700584:

public void dropList1_click1(GDropList source, GEvent event) { //_CODE_:dropList1:776276:
  //println("dropList1 - GDropList >> GEvent." + event + " @ " + millis());
  if (dropList1.getSelectedIndex() == 0){
    //redrawRate = 60;
    historySize = 60;
    //slider1.setLimits(0,speedHistory.size() - historySize);
  }
  if (dropList1.getSelectedIndex() == 1){
    //redrawRate = 300;
    historySize = 60 * 24;
    //slider1.setLimits(0,speedHistory.size() - historySize);
  }
  if (dropList1.getSelectedIndex() == 2){
    historySize = 60 * 24 * 31;
    //slider1.setLimits(0,speedHistory.size() - historySize);
    //redrawRate = 600;
```

```
  }
} //_CODE_:dropList1:776276:

public void textfield4_change1(GTextField source, GEvent event) { //_CODE_:textfield4:410882:
  //println("textfield4 - GTextField >> GEvent." + event + " @ " + millis());
} //_CODE_:textfield4:410882:

public void slider1_change1(GSlider source, GEvent event) { //_CODE_:slider1:276482:
  //println("slider1 - GSlider >> GEvent." + event + " @ " + millis());
  //offset = constrain(slider1.getValueI(), 0, 150);
  offset = slider1.getValueI();
  //if (offset < speedHistory.size()){
  //  textfield8.setText(str(speedHistory.getJSONObject(offset).getFloat("speed")));
  //} else {
  //  textfield8.setText("");
  //}
  //if (offset>temperatures.size()-100){
  //  offset = temperatures.size()-100;
  //  slider1.setValue(offset);
  //}
} //_CODE_:slider1:276482:

public void textfield5_change1(GTextField source, GEvent event) { //_CODE_:textfield5:606495:
  //println("textfield5 - GTextField >> GEvent." + event + " @ " + millis());
} //_CODE_:textfield5:606495:

public void textfield6_change1(GTextField source, GEvent event) { //_CODE_:textfield6:431543:
  //println("textfield6 - GTextField >> GEvent." + event + " @ " + millis());
} //_CODE_:textfield6:431543:

public void textfield7_change1(GTextField source, GEvent event) { //_CODE_:textfield7:681679:
  //println("textfield7 - GTextField >> GEvent." + event + " @ " + millis());
} //_CODE_:textfield7:681679:

public void button1_click1(GButton source, GEvent event) { //_CODE_:button1:328422:
  //println("button1 - GButton >> GEvent." + event + " @ " + millis());
  send = true;
  textfield7.setText("");
} //_CODE_:button1:328422:

// Create all the GUI controls.
// autogenerated do not edit
```

```
public void createGUI(){
  G4P.messagesEnabled(false);
  G4P.setGlobalColorScheme(GCScheme.CYAN_SCHEME);
  G4P.setCursor(ARROW);
  surface.setTitle("Sketch Window");
  label1 = new GLabel(this, 10, 20, 110, 20);
  label1.setText("Текущее время:");
  label1.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  label1.setOpaque(false);
  textfield1 = new GTextField(this, 120, 20, 110, 20, G4P.SCROLLBARS_NONE);
  textfield1.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  textfield1.setOpaque(true);
  textfield1.addEventHandler(this, "textfield1_change1");
  label2 = new GLabel(this, 10, 50, 110, 20);
  label2.setText("Текущая скорость:");
  label2.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  label2.setOpaque(false);
  textfield2 = new GTextField(this, 120, 50, 110, 20, G4P.SCROLLBARS_NONE);
  textfield2.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  textfield2.setOpaque(true);
  textfield2.addEventHandler(this, "textfield2_change1");
  custom_slider1 = new GCustomSlider(this, 240, 40, 230, 40, "grey_blue");
  custom_slider1.setShowValue(true);
  custom_slider1.setShowLimits(true);
  custom_slider1.setLimits(24.0, 0.0, 100.0);
  custom_slider1.setNbrTicks(10);
  custom_slider1.setEasing(20.0);
  custom_slider1.setNumberFormat(G4P.DECIMAL, 0);
  custom_slider1.setOpaque(false);
  custom_slider1.addEventHandler(this, "custom_slider1_change1");
  checkbox1 = new GCheckbox(this, 240, 20, 120, 20);
  checkbox1.setIconAlign(GAlign.LEFT, GAlign.MIDDLE);
  checkbox1.setText("Имитация");
  checkbox1.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  checkbox1.setOpaque(false);
  checkbox1.addEventHandler(this, "checkbox1_clicked1");
  dropList1 = new GDropList(this, 10, 80, 120, 80, 3);
  dropList1.setItems(loadStrings("list_776276"), 0);
  dropList1.setLocalColorScheme(GCScheme.BLUE_SCHEME);
  dropList1.addEventHandler(this, "dropList1_click1");
  textfield4 = new GTextField(this, 10, 110, 120, 20, G4P.SCROLLBARS_NONE);
  textfield4.setLocalColorScheme(GCScheme.BLUE_SCHEME);
```

```
textfield4.setOpaque(true);
textfield4.addEventHandler(this, "textfield4_change1");
slider1 = new GSlider(this, 10, 370, 460, 20, 10.0);
slider1.setLimits(150.0, 0.0, 150.0);
slider1.setNumberFormat(G4P.DECIMAL, 2);
slider1.setOpaque(false);
slider1.addEventHandler(this, "slider1_change1");
textfield5 = new GTextField(this, 10, 350, 120, 20, G4P.SCROLLBARS_NONE);
textfield5.setLocalColorScheme(GCScheme.BLUE_SCHEME);
textfield5.setOpaque(true);
textfield5.addEventHandler(this, "textfield5_change1");
textfield6 = new GTextField(this, 350, 350, 120, 20, G4P.SCROLLBARS_NONE);
textfield6.setLocalColorScheme(GCScheme.BLUE_SCHEME);
textfield6.setOpaque(true);
textfield6.addEventHandler(this, "textfield6_change1");
label4 = new GLabel(this, 240, 90, 100, 40);
label4.setTextAlign(GAlign.LEFT, GAlign.TOP);
label4.setText("Время для синхронизации:");
label4.setOpaque(false);
textfield7 = new GTextField(this, 340, 90, 120, 20, G4P.SCROLLBARS_NONE);
textfield7.setOpaque(true);
textfield7.addEventHandler(this, "textfield7_change1");
button1 = new GButton(this, 340, 120, 120, 20);
button1.setText("Синхронизировать");
button1.addEventHandler(this, "button1_click1");
label5 = new GLabel(this, 5, 330, 20, 20);
label5.setText("0");
label5.setOpaque(false);
label6 = new GLabel(this, 5, 300, 20, 20);
label6.setText("10");
label6.setOpaque(false);
label7 = new GLabel(this, 5, 270, 20, 20);
label7.setText("20");
label7.setOpaque(false);
label8 = new GLabel(this, 5, 240, 20, 20);
label8.setText("30");
label8.setOpaque(false);
label9 = new GLabel(this, 5, 210, 20, 20);
label9.setText("40");
label9.setOpaque(false);
label10 = new GLabel(this, 5, 180, 20, 20);
label10.setText("50");
```

```
  label10.setOpaque(false);
}
// Variable declarations
// autogenerated do not edit
GLabel label1;
GTextField textfield1;
GLabel label2;
GTextField textfield2;
GCustomSlider custom_slider1;
GCheckbox checkbox1;
GDropList dropList1;
GTextField textfield4;
GSlider slider1;
GTextField textfield5;
GTextField textfield6;
GLabel label4;
GTextField textfield7;
GButton button1;
GLabel label5;
GLabel label6;
GLabel label7;
GLabel label8;
GLabel label9;
GLabel label10;
import interfascia.*;
import g4p_controls.*;
//import Serial communication library
import processing.serial.*;
//init variables
Serial commPort;
float speed;
int yDist;
String[] temperature;
int redrawRate = 60; //запись каждую секунду
int historySize = 60;
int offset;
int xSize, ySize, x, y;
JSONArray speedHistory;
boolean startDraw;
String val;
boolean firstContact = false;
boolean firstDraw = false;
```

```
boolean send = false;
void setup()
{
  frameRate(60);
  //setup fonts for use throughout the application
  //set the size of the window
  size(480, 400);
  //init serial communication port
  //commPort = new Serial(this, "COM5", 115200);
  //commPort.bufferUntil('\n');
  createGUI();

  speedHistory = loadJSONArray("wind_speed_history.json");
  background(123);
  startDraw = false;
  xSize = 450;
  ySize = 170;
  x = 20;
  y = 170;
  if (speedHistory.size()>xSize) {
    slider1.setLimits(0, speedHistory.size() - xSize);
    offset = speedHistory.size() - xSize - 1;
  } else {
    slider1.setLimits(0, 1);
    offset = 0;
  }
  slider1.setValue(offset);
}
void draw()
{
  background(123);
  textfield1_change1(null, null);
  //draw graph
  stroke(0);
  fill(255, 255, 255);
  rect(x, y, xSize, ySize);
  if (speedHistory.size()==0) {
    textfield5.setText("");
    textfield6.setText("");
  } else {
    if (speedHistory.size()>xSize) {
      textfield5.setText(speedHistory.getJSONObject(offset).getString("time"));
```

```
  } else {
    textfield5.setText(speedHistory.getJSONObject(0).getString("time"));
  }
  if (speedHistory.size()>offset + xSize) {
    textfield6.setText(speedHistory.getJSONObject(offset + xSize - 2).getString("time"));
  } else {
    textfield6.setText(speedHistory.getJSONObject(speedHistory.size()-1).getString("time"));
  }
}
for (int index = 0; index<xSize; index++)
{
  stroke(0, 0, 0);
  point(x + index, y + ySize - 3 * 10);
  point(x + index, y + ySize - 3 * 20);
  point(x + index, y + ySize - 3 * 30);
  point(x + index, y + ySize - 3 * 40);
  point(x + index, y + ySize - 3 * 50);
  if (offset + index < speedHistory.size()) {
    stroke(255, 0, 0);
    point(x + index+1, y + ySize - 3 * speedHistory.getJSONObject(offset+index).getFloat("speed"));
  }
}
if (firstDraw && startDraw) {
  JSONObject temperatureJ = new JSONObject();
  temperatureJ.setString("time", day()+"."+month()+"."+year()+" "+hour()+":"+minute()+":"+second());
  temperatureJ.setFloat("speed", speed);
  speedHistory.setJSONObject(speedHistory.size(), temperatureJ);
  firstDraw = false;
}
float middleSpeed = 0;
for (int index = 0; index<historySize; index++) {
  if (speedHistory.size()>historySize) {
    if (speedHistory.size()>historySize+offset) {
      middleSpeed = middleSpeed + speedHistory.getJSONObject(index+offset).getFloat("speed");
    } else {
      middleSpeed = middleSpeed + speedHistory.getJSONObject(speedHistory.size()-index-1).getFloat("speed");
    }
  } else {
    if (speedHistory.size()>index)
      middleSpeed = middleSpeed + speedHistory.getJSONObject(index).getFloat("speed");
  }
}
```

```
  middleSpeed = middleSpeed / historySize;
  textfield4.setText(str(middleSpeed));

  if (frameCount % redrawRate == 0 && startDraw) {

    if (checkbox1.isSelected()) {
       speed = custom_slider1.getValueI();
       textfield2_change1(null, null);
       startDraw = true;
     }
    JSONObject speedJ = new JSONObject();
    speedJ.setString("time", day()+"."+month()+"."+year()+" "+hour()+":"+minute()+":"+second());
    speedJ.setFloat("speed", speed);
    speedHistory.setJSONObject(speedHistory.size(), speedJ);

    if (speedHistory.size()>xSize) {
      slider1.setLimits(0, speedHistory.size() - xSize);
    } else {
      slider1.setLimits(0, 1);
    }
    slider1.setValue(offset);
  }
}
void dispose() {
  saveJSONArray(speedHistory, "wind_speed_history.json");
}
```