МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп’ютерної інженерії

ДО ЗАХИСТУ ДОПУЩЕНИЙ
Зав. кафедри
_____

д.т.н., проф. Переверзєв А.В.

ДИПЛОМНА РОБОТА

ТЕМА: «Розробка системи моніторингу стану комп’ютера»

Виконав
ст. гр. КІ–118к9 _____ Є. О. Сілін

Керівник
викладач _____ К. С. Суха

Запоріжжя
2022

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедрою
д.т.н., проф.
А.В. Переверзєв

_____

13.03.2022 р.

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

Студенту гр. *КІ-118к9*, спеціальності *123 «Комп'ютерна інженерія»*

Сілін Євген Олександрович

1.Тема: Розробка системи моніторингу стану комп'ютера

затверджена наказом по інституту 09.2-17 від 12 березня 2022 р.

2. Термін здачі студентом закінченої роботи: 19 червня 2022 р.

3. Перелік питань, що підлягають розробці:

1.Вивчити літературу, присвячену темі розробки.

2. Огляд предметної області та виявлення основних параметрів

майбутньої системи.

3. Огляд існючих аналогів з виявленням переваг та недоліків для релізації власного проекту.

4. Вибір та обгрунтування програмно–апаратного рішення.

5.Виконати монтаж компонентів схеми та налагодити оптимальні режими роботи пристрою

6.Запрограмувати пристрій та відлагодити його

7. Оформити результати роботи у вигляді звіту.

Дата видачі завдання: 13 березня 2022 р.

Керівник випускної роботи _____ К. С. Суха
(підпис)

Завдання отримав до виконання _____ Є.О. Сілін
(підпис студента)

РЕФЕРАТ

Пояснювальна записка складається з: 71 сторінок, 34 рисунка, 1 приложение, 12 джерела.

Об’єкт дослідження – «Системи моніторингу стану комп’ютера на платформі Arduino».

Мета роботи – створення пристрою для зручного спостереження за характеристиками комп’ютера.

Методи дослідження – проектування, моделювання та програмування комп’ютерної системи.

Обрано апаратну частину проекту, що складається з плати Arduino Nano, одного LCD дисплею 2004, датчику температури Ds18b20, комп’ютерного кулера, RGB стрічки, MOSFET транзисторів, резисторів, макетної плати та molex роз'єм для живлення.

Створено програмний код для управління мікроконтроллером.

Створено робочий макет «Системи моніторингу апаратної частини ПК на платформі Arduino», що може бути використано для подальшого застосування на практиці.

ARDUINO, ARDUINO IDE, КОМП’ЮТЕР, МІКРОКОНТРОЛЕР, МОНІТОРИНГ, УТИЛІТА

ЗМІСТ

СПИСОК УМОВИХ ПІЗНАЧЕНЬ

Слово/словосполучення	Скорочення
Широтно–імпульсна модуляція	ШИМ
Програмне забезпечення	ПЗ
Персональний комп’ютер	ПК
Металооксидні напівпровідникові польові транзистори (англ. Metal Oxide Semiconductor Field Effect Transistors)	MOSFET
Червоний, зелений, синій (англ. Red green blue)	RGB
Центральний процесор (англ. central processing unit)	CPU
Графічний процесор (англ. graphics processing unit)	GPU
Оперативна пам'ять (англ. Random–access memory)	RAM
Рідкокристалічний дисплей (англ. liquid crystal display)	LCD
Міжінтегральна схема (англ. Inter–Integrated Circuit)	I²C

ВСТУП

У нас час дуже важлива необхідності ретельного відстеження стану температури заліза. Це пов’язанно з тим, що стабільність роботи комп'ютера загалом, як і довговічність окремих комплектуючих, безпосередньо залежить від своїх робочої температури. Робоча температура у свою чергу пов'язана з комплексом характеристик та властивостей пристроїв: їх енергоспоживанням, тепловиділенням, робочою напругою, частотою. А збільшення частоти викликає збільшення тепловиділення.

Дипломна робота складається з виведення загальної інформації роботи пристроїв ПК безпосередньо для дізнавання температури та завантаженість елементів ПК.

Цей пристрій допоможе вам бачити Рівень завантаження: CPU, GPU, RAM, відеопамяті, температуру у корпусі комп’ютера та запобігти поломці комп'ютера або окремих комплектуючих. Для цього ще буде допомагати відображення навантаження за допомогою RGB стрічки і для запобігання перегріву керування кулерами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА РОЗГЛЯД ПРОЕКТУ «Розробка системи моніторингу стану комп’ютера»

1.1 Огляд загальних характеристик функціонування ПК

Центральний процесор (CPU) — функціональна частина комп'ютера, що призначена для інтерпретації команд.

Рисунок 1.1 – Центральній процесор

Функції:

– обробка даних по заданій програмі шляхом виконання арифметичних і логічних операцій;

– програмне керування роботою пристроїв комп'ютера.

Архітектура системи команд. З погляду програмістів, під архітектурою процесора мається на увазі його здатність виконувати певний набір машинних кодів. Більшість сучасних десктопних процесорів відносяться до сімейства x86, або Intel–сумісних процесорів архітектури IA32 (архітектура 32–бітових процесорів Intel). Її основа була закладена компанією Intel в процесорі i80386,

проте в подальших поколіннях процесорів вона була доповнена і розширена як самою Intel (введені нові набори команд MMX, SSE, SSE2 і SSE3), так і сторонніми виробниками (набори команд EMMX, 3DNow! і Extended 3DNow!, розроблені компанією AMD).

Ядро. В рамках однієї і тієї ж архітектури різні процесори можуть досить сильно відрізнятися один від одного. І відмінності ці утілюються в різноманітних процесорних ядрах, що мають певний набір суворо обумовлених характеристик. Найчастіше ці відмінності втілюються в різних частотах системної шини (FSB), розмірах кешу другого рівня, підтримці тих або інших нових систем команд або технологічних процесах, за якими виготовляються процесори. Нерідко зміна ядра в одному і тому ж сімействі процесорів спричиняє за собою заміну процесорного роз'єму (сокет, англ. socket), з чого витікають питання подальшої сумісності материнських плат. Проте в процесі вдосконалення ядра виробникам доводиться вносити до нього незначні зміни, які не можуть претендувати на "власне ім'я". Такі зміни називаються ревізіями (англ. stepping) ядра і, найчастіше, позначаються цифро–буквеними комбінаціями. Проте в нових ревізіях одного і того ж ядра можуть зустрічатися досить помітні нововведення. Так, компанія Intel ввела підтримку 64–бітової архітектури EM64T в окремі процесори сімейства Pentium 4 саме в процесі зміни ревізії.

Графічний процесор (GPU) — окремий пристрій персонального комп'ютера або ігрової приставки, виконує графічний рендеринг. Сучасні графічні процесори дуже ефективно обробляють і зображують комп'ютерну графіку, завдяки спеціалізованій конвеєрній архітектурі вони набагато ефективніші в обробці графічної інформації, ніж типовий центральний процесор.

Графічний процесор в сучасних відеоадаптерах використовується як прискорювач тривимірної графіки, але в деяких випадках його можна використовувати і для обрахунків (GPGPU). Обчислювальними особливостями в порівнянні із CPU є: архітектура, максимально націлена на збільшення швидкості обчислень текстур та складних графічних об'єктів, обмежений список команд.

Рисунок 1.2 – Графічний процесор

Операти́вна па́м'ять (Random Access Memory) — пам'ять ЕОМ, призначена для зберігання коду та даних програм під час їхнього виконання. У сучасних комп'ютерах оперативна пам'ять переважно представлена динамічною пам'яттю з довільним доступом DRAM.

Рисунок 1.3 – Оперативна пам'ять

Типи. Різні типи модулів оперативної пам'яті

Протилежністю до пам'яті з довільним доступом є пам'ять з послідовним доступом. При довільному доступі, пам'ять організована таким чином, що в будь–яку мить можна отримати значення, записане в будь–якій комірці пам'яті, не переглядаючи інші комірки. При пам'яті з послідовним доступом, яка реалізується, наприклад, на магнітній стрічці, для доступу до певного елемента пам'яті потрібно прокрутити стрічку, зчитуючи інші елементи.

Види пам'яті довільним доступом:

Напівпровідникова статична (SRAM) — комірками є напівпровідникові тригери. Переваги — невелике енергоспоживання, висока швидкодія. Відсутність необхідності проводити «регенерацію». Недоліки — малий обсяг, висока вартість. Нині широко використовується як кеш–пам'ять процесорів у комп'ютерах.

Напівпровідникова динамічна (DRAM) — кожна комірка є конденсатором на основі переходу КМОН–транзистора. Переваги — низька вартість, великий обсяг. Недоліки — необхідність періодичного прочитування і перезапису кожної комірки — т.з. «регенерації», і, як наслідок, зниження швидкодії, велике енергоспоживання. Процес регенерації реалізується спеціальним контролером, встановленим на материнській платі або в центральному процесорі. DRAM зазвичай використовується як оперативна пам'ять комп'ютерів.

Феромагнітна — є матрицею з провідників, на перетині яких знаходяться кільця або біакси, виготовлені з феромагнітних матеріалів. Переваги — стійкість до радіації, збереження інформації при виключенні живлення; недоліки — мала ємність, велика вага, стирання інформації при кожному

читанні. В наш час в такому, зібраному з дискретних компонентів вигляді, не застосовується.

Проте до 2003 року з'явилася магніторезистивна оперативна пам'ять (MRAM) в інтегральному виконанні. Поєднуючи швидкість SRAM і можливість зберігання інформації при відімкненому живленні, MRAM є перспективною заміною типам ROM і RAM. Проте вона приблизно удвічі дорожча за мікросхеми SRAM (при тій же ємності і габаритах).

1.2 Огляд аналогів

1.2.1 MSI AFTERBURNER

MSI AFTERBURNER – найпопулярніша в світі утиліта для моніторинга та повного контролю над відеокартою. Цей додаток надасть максимально повну інформацію про стан вашої системи, дасть змогу зберігати різноманітні комбінації по різним профілям, він має іннегрований бенчмарк та можливість захоплення відеопотоку.

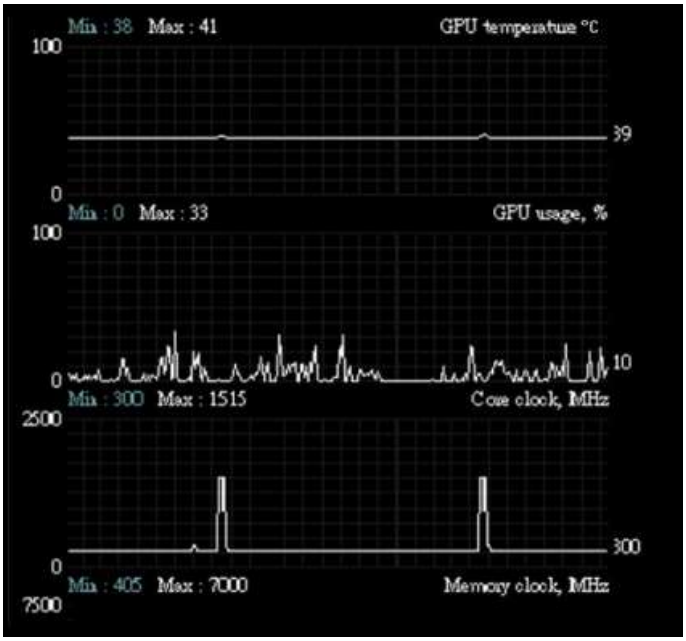


Рисунок 1.4 – Інформація про GPU у MSI AFTERBURNER

1.2.2 Реобас

Реобас – це спрощена назва контролера, що управляє швидкістю руху кулерів та вентиляторів систем охолодження електронного пристрою. Часто такі контролери оснащені додатковими корисними функціями, як моніторинг температури систем і підсистем, і багато іншого залежно від конкретної моделі. Як правило, реобаси встановлюються в порт 5.25", але можливе встановлення і в порт 3.5". Існує велика кількість панелей такого роду – з додатковими виводами USB, аудіовходами та аксесуарами.

Рисунок 1.5 – Реобас DEXP CR–6025U

Пристрій є модуль розмірами 149x100x42мм. Монтується в стандартний відсік системного блоку 5.25". Їм часто замінюють марний оптичний привід. Модель розрахована на керування роботою двох вентиляторів. На фронтальну панель виведено два механічні регулятори для зміни швидкості обертання відповідних вентиляторів.

1.3 Принципова схема функціонування системи

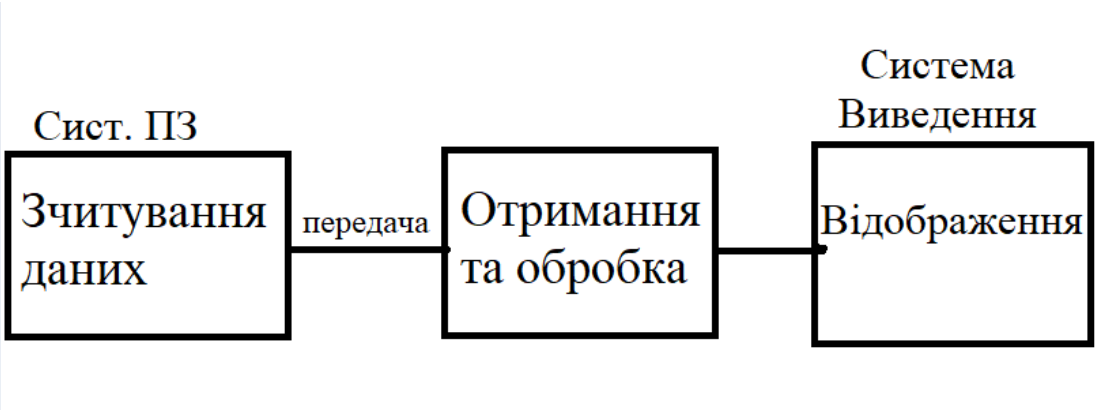


Рисунок 1.5 – Принципова схема функціонування системи

Схема функціонування системи складається з зчитування з зчитування даних з комп’ютера та передачу до Arduino, там данні обробляються та відображаються на дисплеї, а також навантаження передається через колір на RGB стрічці.

1.4 Висновки за розділом

Здійснивши огляд предметної області створення системы моніторингу апаратної частини ПК встановлено, що створення цього монітору є гарним рішенням для модифікування свого ПК, для спостереження за станом температури, керуванням та використанням та коригуванням швидкості кулерів комп'ютера. та для покращення своїх навичок в програмуванні. Для покращення спостереження за температурою додано РГБ стрічку. Ця інформація важлива для того, щоб дізнатися при перегріві комплектуючих.

РОЗДІЛ 2
ВИБІР ТА ОБҐРУНТУВАННЯ ПРОГРАМНО–АПАРАТНОГО КОМПЛЕКСУ

2.1 Вибір мікроконтролеру та платформи

Nano – одна з найбільш мініатюрних плат Arduino. Вона є повним аналогом Arduino Uno – так само працює на чіпі ATmega328P (хоча можна ще зустріти варіанти з ATmega168), але з меншим форм–фактором. Через своїх габаритних розмірів плата часто використовується в проектах, в яких важлива компактність. На платі відсутня винесене гніздо зовнішнього живлення, Arduino працює через USB (miniUSB або microUSB). В іншому параметри збігаються з моделлю Arduino Uno.

Рисунок 2.1 – Arduino NANO V3

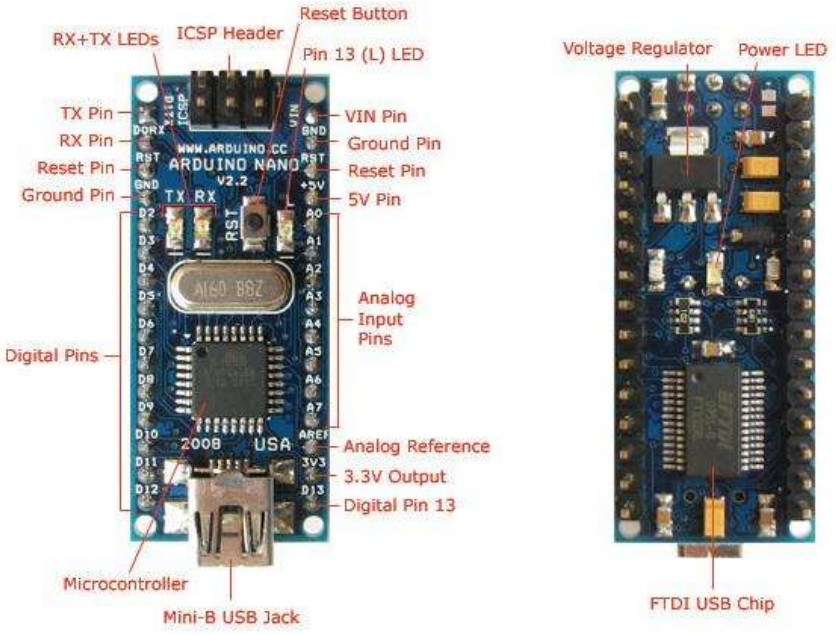


Рисунок 2.2 – Опис плати Arduino NANO

Технічні характеристики Arduino Nano:

- напруга живлення 5В;
- вхідне живлення 7–12В (рекомендований);
- кількість цифрових пинов – 14, з них 6 можуть використовуватися в якості виходів ШІМ;
- 8 аналогових входів;
- максимальний струм цифрового виходу 40 мА;
- флеш пам'ять 16 Кб або 32 Кб, в залежності від чіпа;
- ОЗУ 1 Кб або 2 КБ, в залежності від чіпа;
- EEPROM 512 байт або 1 Кб;
- частота 16 МГц;
- розміри 19 х 42 мм;
- вага 7 м;

Живлення плати може здійснюватися двома способами:

1. Через mini–USB або microUSB при підключенні до комп'ютера;
2. Через зовнішнє джерело живлення, що має напругу 6–20 В з низьким рівнем пульсацій.

Стабілізація зовнішнього джерела виконується за допомогою схеми LM1117IMPX–5.0 на 5В. При підключенні через кабель від комп'ютера підключення до стабілізатора відбувається через діод Шотткі. Схеми обох типів харчування наведені на малюнку.

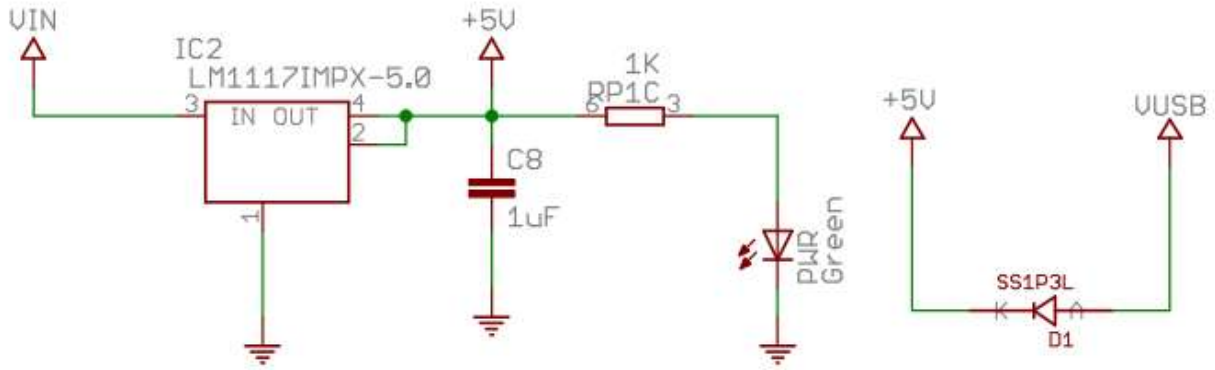


Рисунок 2.3 – Схема Arduino NANO

При підключенні двох джерел напруги плата вибирає з найбільшим харчуванням.

У плати Arduino Nano є такі ж обмеження по напрузі і струму на входи і виходи плати. Всі цифрові і аналогові контакти працюють в діапазоні від 0 до 5 В. При подачі живлення, що виходить за рамки цих значень, напруга буде обмежуватися захисними діодами. У цьому випадку сигнал повинен підключатися через резистор, щоб не вивести контролер з ладу. Найбільше значення впадає або випливає струму не повинно перевищувати значення 40 мА, а загальний струм контактів повинен бути не більше 200 мА.

На платі є 4 світлодіода, які показують стан сигналу. Вони позначені як TX, RX, PWR і L. На перших двох світлодіод загоряється, коли рівень сигналу

низький, і показує, що сигнал TX або RX активний. Світлодіод PWR загоряється при напрузі в 5 В і показує, що підключено харчування. Останній світлодіод – загального призначення, загоряється, коли подається високий сигнал.

На даний момент випускається декілька видів Arduino Nano. Є версії 2.X, 3.0., Які відрізняються тільки чіпом, на якому вони працюють. У версії 2.X. використовується чіп ATmega168 з меншим об'ємом пам'яті (флеш, незалежній) і зниженою тактовою частотою, версія 3.0. працює на чіпі ATmega328.

Терморегулятори Arduino Nano. Плата Arduino Нано має 14 цифрових контактів, які позначаються літерою D (цифровий, digital). Контакти використовуються як входи і виходи, у кожного є підтягаючий резистор.

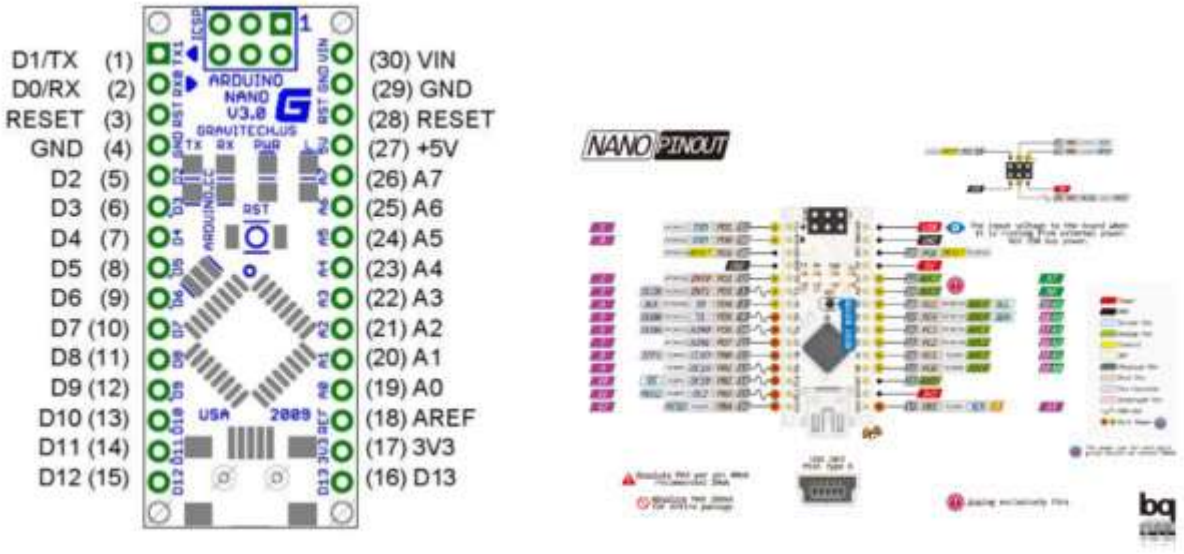


Рисунок 2.4 – Показ контактів і пінів в Arduino NANO

Аналогові Піни позначаються літерою A і використовуються як входи. У них отсутствую підтягує резистори, вони вимірюють подане на них напруга і повертають значення за допомогою функції analogRead ().

На деяких цифрових пінах можна побачити значок ~. Такі контакти можуть бути легко використовувати в якості виходів ШІМ. Arduino нано оснащена шістьма такими контактами – це Піни D3, D5, D6, D9, D10, D11. Для роботи з ШІМ висновками використовується функція analogWrite (). Опис пинов Arduino Нано:

- цифрові входи / виходи: D0–D13;
- аналогові входи / виходи: A0–A7 (10–розрядний АЦП);
- ШІМ: Піни 3, 5, 6, 9, 10, 11;
- UART: D0 і D1 (TX і RX відповідно);
- I2C: SDA – A4, SCL –A5;
- SPI: MOSI – 11, MISO – 12, SCK – 13, SS (10);

Пробіжить по пінам:

- 0 – TX (передача даних UART), D0;
- 1 – RX (прийом даних UART), D1. RX і TX можуть використовуватися для зв'язку по послідовному інтерфейсу або як звичайні порти даних;
- 3, 29 – скидання;
- 4, 29 – земля;
- 5 – D2, переривання INT0;
- 6 – D3, переривання INT1 / ШІМ / AIN0;
- 7 – A4, лічильник T0 / шина I2C SDA / AIN1. AIN0 і AIN1 – входи для швидкодіючого аналогового компаратора;
- 8 – A5, лічильник T1 / шина I2C SCL / ШІМ;
- 9 – 16 – порти D6–D13, з яких D6 (9й), D9 (12й), D10 (13й) і D11 (14й) використовуються як виходи ШІМ. D13 (16й пін) – світлодіод. Також D10 – SS, D11 – MOSI, D12 – MISO, D13 – SCK використовуються для зв'язку по інтерфейсу SPI;

- 18 – AREF, це опорна напруга для АЦП мікроконтролера;
- 19 – 26: аналогові входи A0 ... A7. Розрядність АЦП 10 біт. A4 (SDA), A5 (SCL) – використовуються для зв'язку по шині I2C. Для створення використовується спеціальна бібліотека Wire;

Мікроконтролери мають великі функціональні можливості, але у них є один недолік – це обмежене, по сраувенію з Arduino Mega, число висновків. Тому на етапі складання схеми пристрою слід продумати, яким чином можна максимально спростити проект, щоб скоротити число потрібних для підключення контактів.

Підключення Arduino Nano. Підключення плати Arduino Nano до комп'ютера не становить особливих труднощів – воно аналогічно звичайній платі Uno. Єдина складність може виникнути при роботі з платою на базі чіпа ATMEGA 168 – в настройках потрібно вибрати спершу плату Nano, а потім потрібний варіант процесора.

Установка драйвера для CH340. Мікросхема CH340 часто використовується в платах Arduino з вбудованим USB–to–Serial перетворювачем. Вона дозволяє зменшити витрати на виробництво плат, не впливаючи на її працездатність. За допомогою цього програматора можна легко прошивати плати Arduino. Для того, щоб почати працювати з цією мікросхемою, потрібно встановити драйвер на комп'ютер. Установка виконується в кілька етапів:

- завантаження архіву з драйвером для потрібної операційної системи. Для Windows, MacOS і Linux завантажити драйвери можна за посиланням в нашій статті про USB UART;
- розпакування архіву;
- пошук файлу SETUP.EXE, його запуск;

- на моніторі з'явиться вікно, в якому потрібно натиснути кнопку Install. Установка драйвера почнеться, після чого можна починати роботу зі схемою;

I²C (Inter–Integrated Circuit) — послідовна асиметрична шина для зв'язку між інтегральними схемами всередині електронних приладів . Використовує дві двонаправлені лінії зв'язку (SDA і SCL), застосовується для з'єднання низько швидкісних периферійних компонентів з процесорами і микроконтроллерами. Даний модуль дозволяє зменшити кількість використовуваних висновків контролера, замість 8 або 4–бітного з'єднання, потрібно тільки 2 виведення (SDA і SCL).

Рисунок 2.5 – модуль I2C

Технічні характеристики:

- Підтримка дисплеїв: LCD 16×02/20×04;
- Додатково: регулювання контрастності;
- Напруга харчування 5В;

- Інтерфейс: I2C;
- Габарити: 54мм х 19мм х 15мм.

Загальні відомості интерфейсного модуля I2C: Оскільки кількість контактів на контролерах Arduino обмежена і часто при використанні різних датчиків і модулів вони закінчуються, з'являється необхідність в їх економії, для цих випадок розроблений цей модуль, з його допомогою можна реалізувати передачу по двом контактам (SDA і SCL).

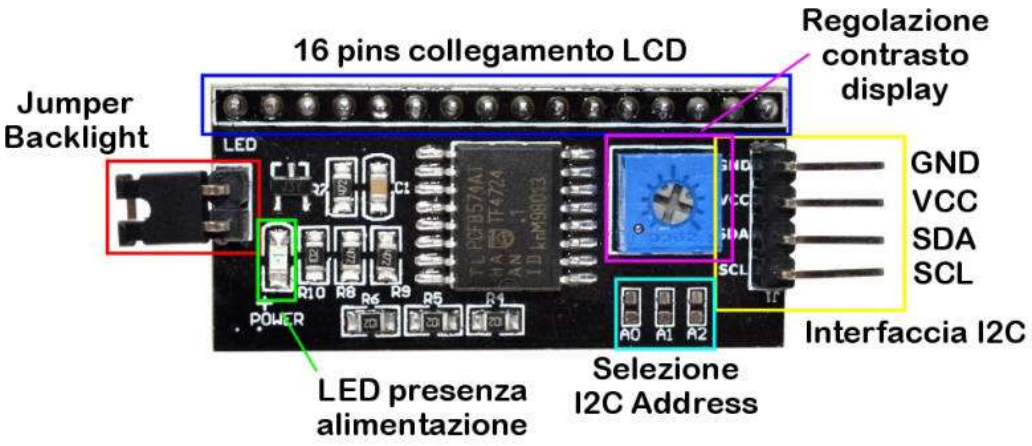


Рисунок 2.6 – Група контактів в I2C

Тепер трохи про самому модулі, побудований він на мікросхемі PCF8574T. Резистори R8 (4.7кОм) і R9 (4.7кОм) необхідні для підтяжки ліній SDA і SCL, в ідеалі при підключенні двох і більше пристроїв по шині I2C необхідно використовувати підтяжку тільки на одному пристроїв. На платі передбачені три перемички (за схемою видно що лінії A0, A1, A2 підтягнуті до харчування через резистори R4, R5, R6), необхідні вони для зміни адресації пристрої, всього їх 8 варіантів. Зміна адресації дає нам можливість підключення до восьми пристроїв по шині IC2 с мікросхемою PCF8574T, варіанти адрес показані на малюнку (за замовчуванням адреса пристрою 0x27). Так само

модуль оснащений потенціометром R11 з його допомогою можна змінити контрастність LCD дисплея.

A2	A1	A0	Адрес
-	-	-	0x20
-	-	+	0x21
-	+	-	0x22
-	+	+	0x23
+	-	-	0x24
+	-	+	0x25
+	+	-	0x26
+	+	+	0x27

Рисунок 2.7 – Лінії A0, A1, A2

Для з'єднання на модулі розташовано три групи контактів:

Перша група**:**

– SCL: лінія тактирования (Serial CLock);

– SDA: лінія даних (Serial Data);

– VCC: «+» харчування;

– GND: «–» харчування;

Друга група**:**

– VSS: «–» харчування;

– VDD: «+» харчування;

– VO: Висновок управління контрастом;

– RS: Вибір регістру;

– RW: Читання/запись(режим запису приьз'єднанні з землею);

– E: Enable (стрибає по спаду);

– DB0–DB3: Молодші біти інтерфейсу;

– DB4–DB7: Старші біти інтерфейсу;

– A: «+» харчування підсвічування;

– K: «–» харчування підсвічування;

Третя група: (за замовчуванням встановлена перемичка)

– VCC:

– A від LCD:

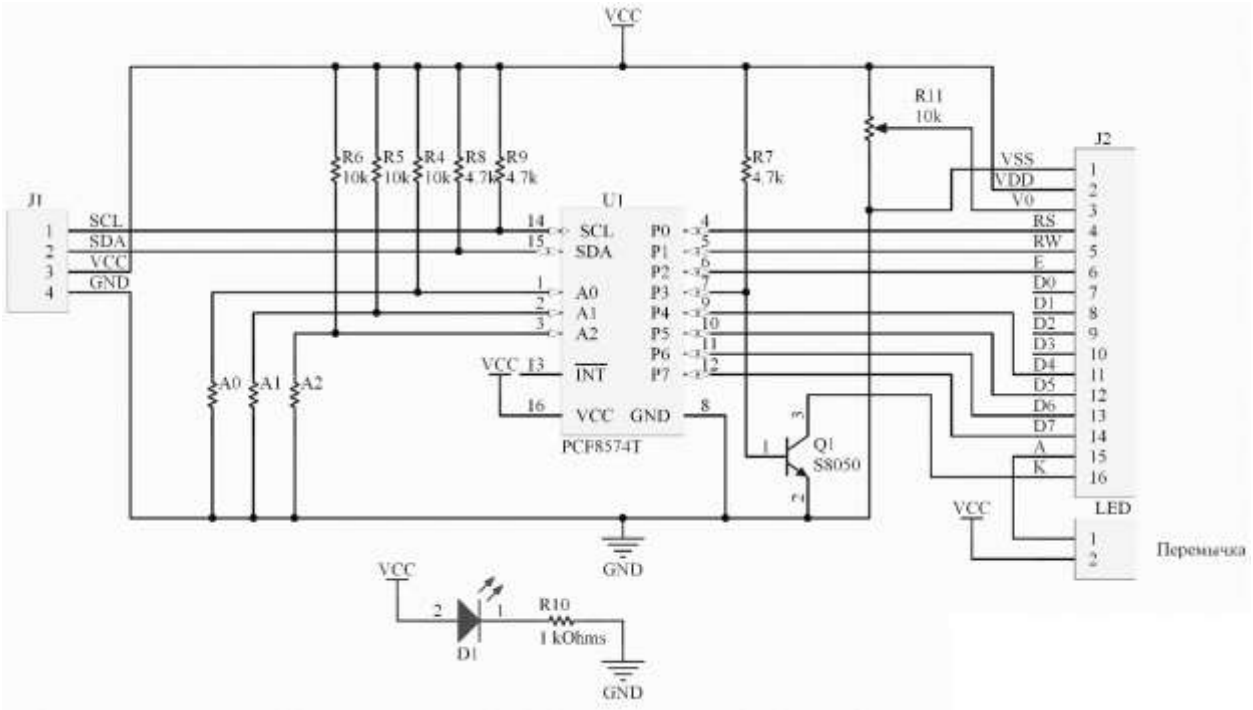


Рисунок 2.8 – Інтерфейсний модуль I2C

Можна виділити кілька переваг:

- для роботи потрібно всього 2 лінії – SDA (лінія даних) і SCL (лінія синхронізації);
- підключення великої кількості провідних приладів;

- зменшення часу розробки;
- для управління всім набором пристроїв потрібно тільки один мікроконтролер;
- можливе число підключаються мікросхем до однієї шині обмежується тільки граничної ємністю;
- високий ступінь збереження даних через спеціального фільтра переважної сплески, вбудованого в схеми;
- проста процедура діагностики виникають збоїв, швидка налагодження несправностей;
- шина вже інтегрована в саму Arduino, тому не потрібно розробляти додатково шинний інтерфейс;

Недоліки:

- існує ємнісне обмеження на лінії – 400 пФ;
- важке програмування контролера I2C, якщо на шині є кілька різних пристроїв;
- при великій кількості пристроїв виникає труднощі локалізації збою, якщо одне з них помилково встановлює стан низького рівня;

Рідкокристалічний дисплей (Liquid Crystal Display) LCD 2004 є хорошим вибором для виведення рядків символів в різних проектах. Він коштує недорого, є різні модифікації з різними кольорами підсвічування, ви можете легко завантажити готові бібліотеки для скетчів Arduino.

Короткий опис пинов LCD

Давайте подивимося на висновки LCD 2004 уважніше:

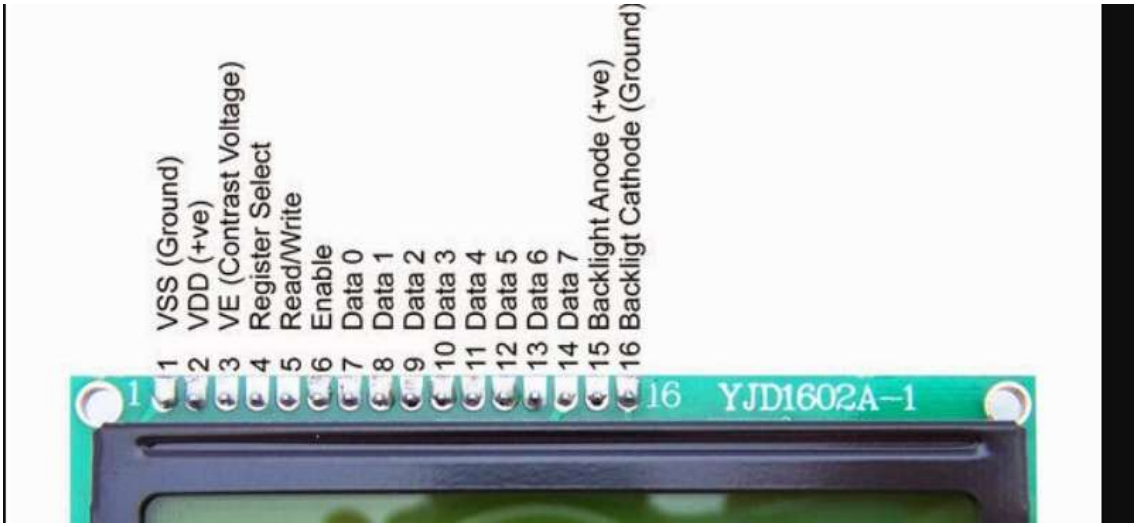


Рисунок 2.9 – Короткий опис пінов LCD 2004

Кожен з виводів має своє призначення:

1. Земля GND;
2. Живлення 5 В;
3. Установка контрастності монітора;
4. Команда, дані;
5. Записування та читання даних;
6. Enable;

7–14. Лінії даних;

15. Плюс підсвічування;
16. Мінус підсвічування.

Технічні характеристики дисплея:

- символьний тип відображення, є можливість завантаження символів;
- світлодіодна підсвітка;
- контролер HD44780;
- напруга живлення 5В;

- формат 20x4 символів;
- діапазон робочих температур від –20С до + 70С, діапазон температур зберігання від–30С до +80 С;
- кут огляду 180 градусів;

Схема підключення LCD до плати Arduino без i2C

Стандартна схема приєднання монітора безпосередньо до мікроконтролера Arduino без I2C виглядає наступним чином.

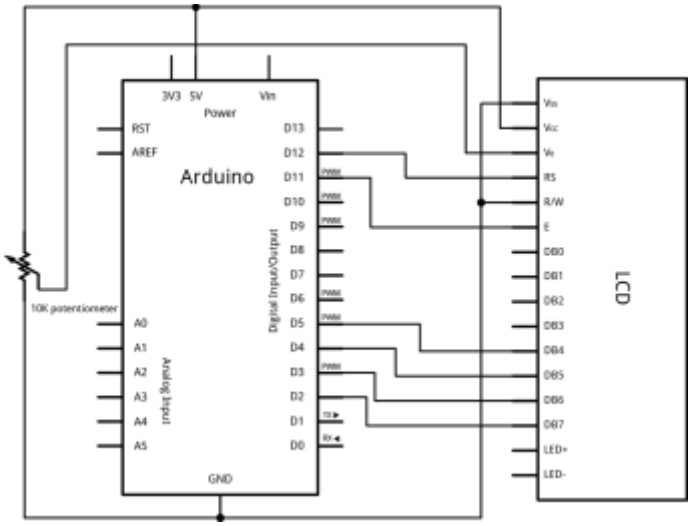


Рисунок 2.10 – Приєднання монітора до мікроконтролера Arduino без I2C

Через велику кількість підключаються контактів може не вистачити місця для приєднання потрібних елементів. Використання I2C зменшує кількість проводів до 4, а зайнятих пинов до 2.

Підключення до Arduino. Необхідні деталі:

– Arduino UNO R3 x 1 шт;

– LCD–дисплей 1602A (2 × 16, 5V, Синій) x 1 шт;

– інтерфейсний модуль I2C, IIC, TWI для LCD x 1 шт;

– провід DuPont, 2,54 мм, 20 см, FM (Female – Male) x 1 шт;

– кабель USB 2.0 AB x 1 шт;

Підключення :

Насамперед, припаюємо модуль I2C до LCD дисплеєм, а потім необхідно підключити дисплей до Arduino UNO. Для цього скористаємося проводками DuPont, підключення здійснюємо по таблиці нижче.

Другу частину розробки буде зібрано на макетній платі.

Макетна плата – універсальна друкована плата для збирання та моделювання прототипів електронних пристроїв. Макетні плати поділяються на два типи: для монтажу за допомогою пайки та без такої.

Є кілька різних типів макетних плат:

Універсальні мають виключно металізовані отвори, які розробник повинен з'єднувати перемичками.

Для цифрових пристроїв намічені можливі місця під мікросхеми, по всій платі проведені шини живлення.

Спеціалізовані – для пристроїв на мікросхемі конкретної моделі. На таких платах є як розведені стандартні ланцюги, так і матриця отворів і доріжок для нестандартних. Наприклад, для мікроконтролерів стандартними ланцюгами будуть:

- посадкове місце для мікросхеми;
- живлення;
- «земля»;
- кварцовий резонатор та лінії внутрішньосхемного програмування.

Для диплому було обрано порожню двосторонню макетну палату 5x7см. Тип макетної плати МАС–1.

Рисунок 2.11 – Макетна плата PCB–5x7

Для того щоб знати температуру всередині корпусу використовується датчик температури DS18B20.

DS18B20 – це датчик температури, який має роздільну здатність перетворення від 9 до 12 розрядів. Тривожний сигнал – функція, яка дозволяє якісно контролювати температуру рідини. Більшість параметрів контролю задаються самостійно користувачем. Вони зберігаються у пам'яті і можуть бути перенастроєні у майбутньому. Датчик ds18b20 використовує протокол 1–Wire інтерфейсу для обміну даними.

Рисунок 2.12 – Датчик ds18b20

Датчик може вимірювати температуру в широкому діапазоні, від –55 до +125 градусів за Цельсієм. Похибка мінімальна і найчастіше становить максимум півградуса. Вищеописані характеристики роблять датчки популярним для використання в екологічному контролі, моніторингу температурних змін у будинках, а також у вузлах обладнання.

Короткий огляд характеристик:

Датчик дозволяє вносити коригування конфігурацію, використовуючи регістр. Можна налаштувати параметри перетворення агрегату. Цифра варіюється від 9 до 12 біт. Вся інформація буде внесена в незалежну пам'ять, яку ще називають (EEPROM). Обмін даними здійснюється спеціальним протоколом 1–Wire. Власнику потрібен невеликий резистор, що підтягує, адже всі агрегати підключаються до загальної шини.

Уточнення:

- кожен елемент підключено до однієї шини;
- спеціалізований протокол ідентифікує кожен пристрій на шині та обмінюється інформацією;
- адреса датчика є в кожному агрегаті, саме він дозволяє мікроконтролеру визначати його та надсилати дані через 64–розрядний код;
- схема підключення виглядає саме так, кількість датчиків може бути необмеженою;

Також датчик може працювати без зовнішніх джерел живлення. Використовується спеціальний резистор та висновок DQ. Сигнали підвищеного рівня заряджають внутрішні конденсатори. Цей метод називається паразитним харчуванням. Яким варіантом харчування користуватись вирішувати саме власнику.[10]

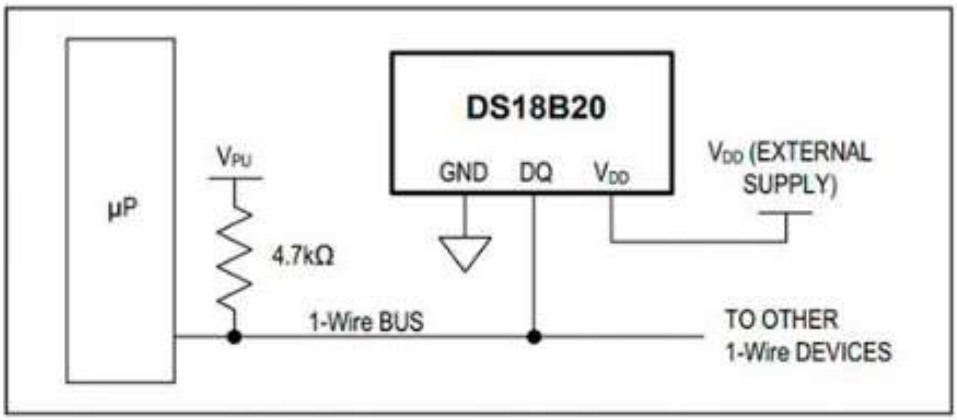


Рисунок 2.13 – Схема підключення датчик ds18b20

Для зручності позначення температури та для декору використовуватиметься RGB стрічка.

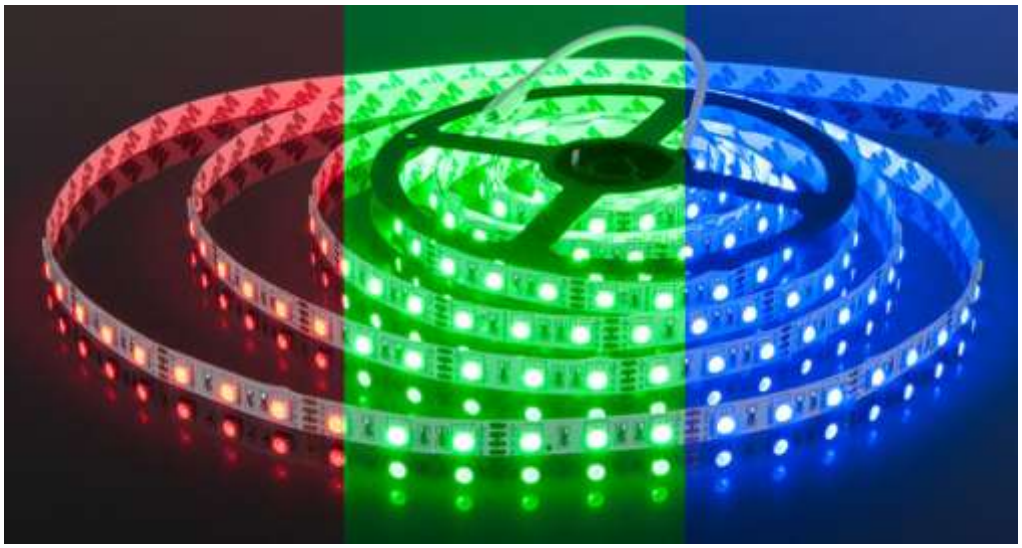


Рисунок 2.14 – RGB стрічка

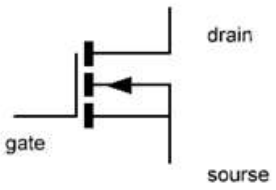
RGB (Red, Green, Blue – червоний, зелений, синій) – це світлодіодна стрічка, яка змінює колір світіння під час роботи. У кожному LED модулі знаходяться три світлодіоди – червоний, синій та зелений. Змінюючи яскравість світіння кожного кристала, можна отримати будь–який колір видимого спектра.[12]

Для підключення РГБ стрічки та датчика температури використовуватиметься MOSFET транзистор. У дипломі використовується MOSFET транзистор IRF3205PBF.

Рисунок 2.15 – MOSFET транзистор IRF3205PBF

MOSFET – це скорочення від двох англійських словосполучень: Metal–Oxide–Semiconductor (метал – оксид – напівпровідник) та Field–Effect–Transistors (транзистор, керований електричним полем). Тому MOSFET – це нічим іншим, як звичайний МОП–транзистор.

Суть роботи польового транзистора полягає в можливості управління струмом, що протікає через нього, за допомогою електричного поля (напруги). Цим він вигідно відрізняється від транзисторів біполярного типу, де керування великим вихідним струмом здійснюється за допомогою малого вхідного струму.[8]



n-канальний MOSFET

Рисунок 2.16 – схема MOSFET транзистора

А також для підключення використовуються резистор 10 кОм та 100 Ом.

Рисунок 2.17 – резистор 10 Ом

Резистор – пасивний елемент електричного кола. Також його називають "опір", завдяки здатності обмежувати струм, створюючи для нього перешкоду.

Резистори використовуються практично у всіх електричних схемах. Найчастіше їх використовують для поділу або зменшення напруги, керування силою струму.[9]

Для переміщення між етапами на дисплеї використовуються кнопки B3F–1070

Характеристики:

Тип контактних груп OFF–(ON), максимальний струм (Imax), мА 50, тип монтування ТНТ, тип контактної групи (США) SPST.

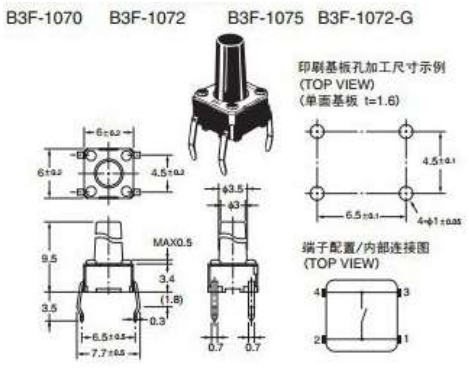


Рисунок 2.18 – кнопка B3F–1070

Для підключення вентиляторів використовуються гвинтовий клеммник DG306–5.0–02P–12–00AH

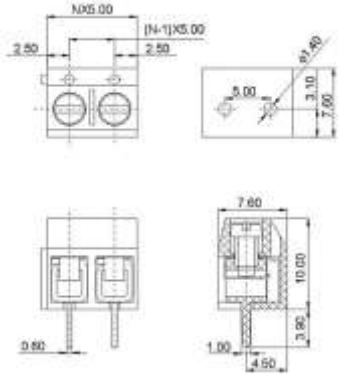


Рисунок 2.19 – клеммник DG306–5.0–02P–12–00AH

Для живлення цієї системи вмонтовується кабель Molex 12В

Рисунок 2.20 – Molex кабель

2.2 Вибір середовища розробки та мови програмування

Не менш важливим питанням що до вирішення є використання програмних засобів для проектування системи, написання відповідного набору команд та прошивання платформи. Тому розглянемо основні програмні компоненти нижче.

Налаштування Arduino IDE

Стандартна середовище розробки Arduino IDE використовується для роботи всіх видів Arduino з комп'ютером. Щоб почати роботу, потрібно спочатку завантажити Arduino IDE з офіційного сайту і встановити її. Зручніше завантажувати Windows Installer, особливо якщо середовище розробки буде встановлена на постійному робочому комп'ютері. Якщо викачаний архів, то його потрібно розпакувати і запустити файл Arduino.exe.

Як тільки середовище встановлена, потрібно її запустити. Для цього потрібно підключити до комп'ютера саму плату Arduino через USB. Потім перейти в меню Пуск >> Панель управління >> Диспетчер пристроїв, знайти там Порти COM і LPT. У списку з'явиться встановлена плата і вказано номер порту, до якого підключається плата.

Після цього потрібно запустити Arduino IDE, перейти в меню Інструменти >> Порт, і вказати порт, до якого приєднана Arduino. У мене Інструменти >> Плати потрібно вибрати модель підключеної плати, в даному випадку Arduino Nano. Якщо у вас плата Nano версії 2.0, то вам потрібно також вибрати варіант процесора в відповідному меню.

Важливо пам'ятати, що якщо до комп'ютера буде підключатися інша плата, настройки знову потрібно буде поміняти на відповідний пристрій.

2.2.1 Середовище розробки

Arduino – це електронний конструктор і зручна платформа швидкої розробки електронних пристроїв для новачків і професіоналів. Платформа користується величезною популярністю в усьому світі завдяки зручності і

простоті мови програмування, а також відкритої архітектури і програмного коду. Пристрій програмується через USB без використання программаторов.

Arduino дозволяє комп'ютеру вийти за рамки віртуального світу в фізичний і взаємодіяти з ним. Пристрої на базі Arduino можуть отримувати інформацію про навколишнє середовище за допомогою різних датчиків, а також можуть управляти різними виконавчими пристроями.

Микроконтроллер на платі програмується за допомогою мови Arduino (заснований на мові Wiring) і середовища розробки Arduino (заснована на середовищі Processing). Проекти пристроїв, засновані на Arduino, можуть працювати самостійно, або ж взаємодіяти з програмним забезпеченням на комп'ютері (наприклад, Flash, Processing, MaxMSP). Плати можуть бути зібрані користувачем самостійно або куплені в зборі.

Середовище розробки Arduino складається з вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів з кнопками часто використовуваних команд і декількох меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino. Програма, написана в середовищі Arduino, називається скетч. Скетч пишеться в текстовому редакторі, що має інструменти вирізки / вставки, пошуку заміни тексту. Під час збереження і експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися виникли помилки.

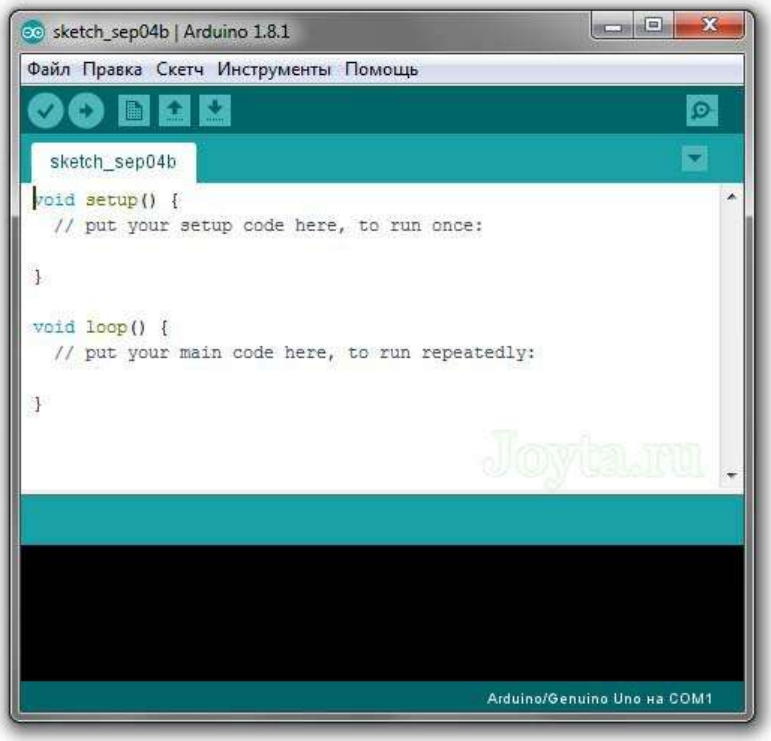


Рисунок 2.21 – Інтерфейс Arduino IDE

В Arduino IDE можна знайти основні функціональні елементи:

1) меню програми;
2) панель швидкого доступу до найбільш важливих функцій;
3) редактор (для розміщення коду програми);
4) панель повідомлень і статусу програми.

Меню програми дозволяє здійснювати управління проектом, наприклад, створення нового проекту, збереження поточного, роздрукувати на принтері вихідний код.

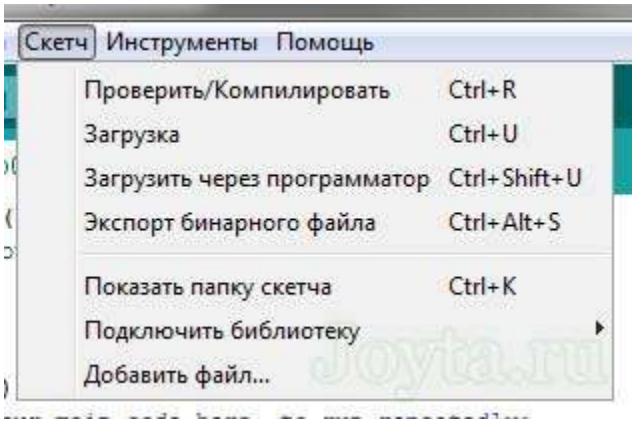


Рисунок 2.22 – Меню «Скетч»

Цікавим і корисним елементом IDE є меню «Інструменти», яке включає до себе функції автоматичного форматування коду, архівування проекту, включення монітора послідовного порту (USB в Arduino розглядається як звичайний послідовний порт).

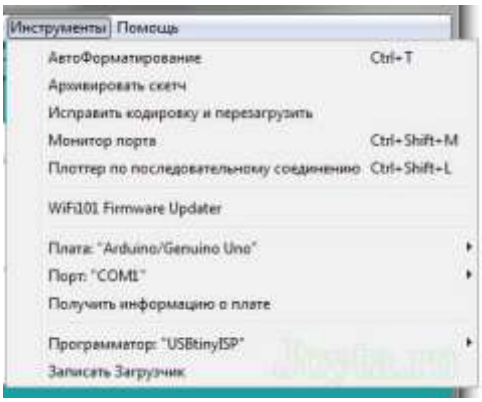


Рисунок 2.23 – Меню «Інструменти»

Найбільш важливим елементом меню «Інструменти» є можливість вибору відповідної плати, тобто вашої системи Arduino підключеної до комп'ютера. У списку знаходяться всі офіційні версії Arduino.

У меню «Інструменти» ви також можете встановити порт, до якого підключена плата Arduino. Пакет Arduino IDE сам визначає порт, але іноді потрібно вручну встановити номер порту в налаштуваннях.

Для нормальної роботи з Arduino IDE використовується панель швидкого доступу, яка оснащена найбільш важливими кнопками. Це рішення, що полегшує роботу з пакетом IDE, дає нам прямий доступ до практично всіх необхідних параметрів при написанні і тестуванні програми.

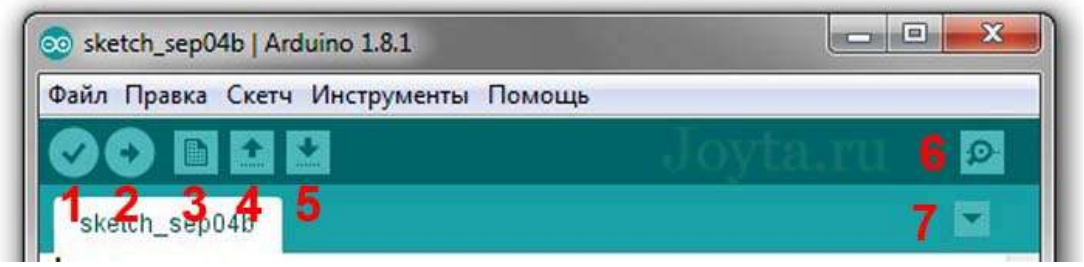


Рисунок 2.24 – Панель швидкого доступу

Вони дозволяють (зліва направо):

- компілювати програму;
- завантажити програму в мікроконтролер (перед прошивкою код програми компілюється);
- почати роботу над новим проектом;
- відкрити існуючий проект;
- зберегти проект на диск;
- включити монітор послідовного порту.

Всі опції, розташовані на панелі швидкого доступу, продубльовані в меню програми.

Останнім елементом програми є вікно повідомлень і статусу. Видима там інформація дозволяє користувачеві знайти помилки в програмному коді і

отримати підтвердження про завершення компіляції і завантаження програми в мікроконтролер.

Arduino IDE в порівняні з її аналогами такими як WinAVR, AVR Studio, добре підходить для новачків, які тільки починають розбиратися в програмуванні та схемотехніці. В самій IDE є дуже багато різних бібліотек розташованих на вершині AVR LibC також має розумний інтерфейс і значно спрощує створення/завантаження.

Платформа Arduino в основному спрощує те, що ви хочете зробити зі своєю схемою, а не займатися технічними тонкощами використання самого мікропроцесора.

Підводячи підсумок можна сказати, що Arduino IDE – це простий програмний пакет, який дозволяє запрограмувати будь–яку відому плату Arduino, спілкуватися з послідовним портом і легко управляти проектами.[6]

2.2.2 Вибір мови програмування

На сьогоднішній час для створення команд для мікроконтролера AVR є Assembler та GCC C/C++. Однак при використанні IDE доречно використовувати мову Processing з компілятором GCC C, що інтегровано до відповідного середовища.

Мова програмування пристроїв Arduino заснований на C / C ++ і скомпонований з бібліотекою AVR Libc і дозволяє використовувати будь–які її функції.

Лістинг 2.1 – Необхідні блоки скетча

```
void setup() {/* відкриваюча дужка на початку програми ініціалізації*/
/* Ця програма виконується 1 раз і призначена для приведення системи у робочий стан, сюди можна записувати функції ініціалізації портів і периферійних пристроїв*/
/* закриваюча дужка в кінці програми ініціалізації*/
void loop() {/* відкриваюча дужка на початку основного циклу*/
/* по суті нескінченний цикл, це основний цикл програми, всередині якого виконується її основна частина(крім початкової ініціалізації)*/
}/* закриваюча дужка в кінці основного циклу*/
```

Функція – структурна одиниця програми, яка має ім'я і містить в собі деяку послідовність дій. Більшість записів в програмі повинні закінчуватися ";", виключення складають оператори циклів, вибору, умов, а також опис прототипів функцій. На початку програми, перед функцією setup, зазвичай, оголошуються змінні і визначення. Після включення живлення плати першої виконується функція setup. Вона виконується тільки один раз. Зазвичай в ній не започатковано режими роботи портів: порти, до яких підключені різні датчики, встановлюються як входи, а порти з виконавчими пристроями як виходи.

Код, написаний в функції loop() починає виконуватися після виконання функції setup(), і виконується в нескінченному циклі знову і знову. У цій функції виконується основна робота: різні обчислення, отримання значень датчиків, виведення значень на порти.

Open Hardware Monitor – безкоштовна утиліта з відкритим вихідним кодом, яка надає централізований інтерфейс, де можна легко контролювати різні аспекти продуктивності обладнання, температурні датчики , споживання

напруги, навантаження і тактові частоти процесора. Це ПО сумісно з більшістю мікросхем, якими обладнуються сучасні популярні плати.

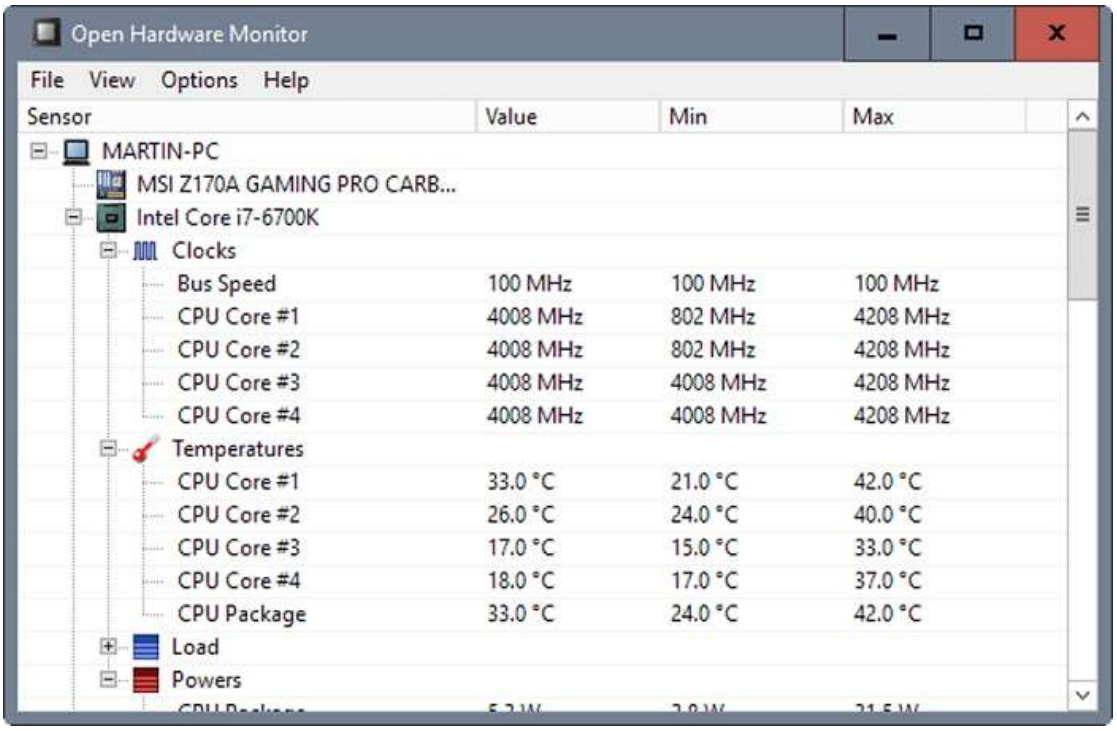


Рисунок 2.25 Утиліта Open Hardware Monitor

Особливості:

Open Hardware Monitor – повністю безкоштовний програмний пакет, призначений для того, щоб користувачі могли контролювати багато аспектів своєї операційної системи в режимі реального часу. Мало того, що це може бути корисно для розробників, такий софт ідеальний для всіх, хто хоче збільшити операційний здатність своєї системи або отримати доступ до додаткових рішень щодо усунення неполадок. Основна мета Open Hardware Monitor – відобразити всі найбільш важливі фізичні показники операційної системи. На регулярній основі вимірюються температура процесора, напруга, що споживаються всіма жорсткими дисками, Швидкість будь–яких вбудованих охолоджувальних вентиляторів і т. Д. Ці фактори можуть допомогти підвищити ефективність

жорсткого диска і збільшити загальний термін служби комп'ютера. Функціональність «в один клік» робить використання програми максимально простим. Утиліту також можна налаштувати на автоматичний запуск в фоновому режимі.

Переваги:

- може запускатися під час запуску системи;
- сумісність з більшістю контрольних мікросхем сучасних плат, в тому числі Winbond, ITE і Fintek;
- дає користувачеві можливість відображати контрольовані значення трьома різними способами – в головному вікні інтерфейсу, за допомогою персоналізованого настроюється гаджета для робочого столу і в системному треї;
- моніторинг і відображення температури процесора процесорів Intel і AMD ;
- моніторинг датчиків відеокарт ATI і Nvidia;
- забезпечує безперервну зворотний SMART (самоконтроль, аналіз і звітність) зв'язок практично для всіх комп'ютерних жорстких дисків;
- визначає критичні значення напруги процесора Vcore і батареї;
- відображає температуру системи і процесора в градусах Цельсія і в Фаренгейті, а також швидкість обертання верхнього, нижнього і заднього вентиляторів, а також джерела живлення.

Недоліки:

1. Середньому користувачеві комп'ютера може бути складно зрозуміти, що означають різні статистичні дані. Тому для ефективного аналізу результатів звітів, створених програмним забезпеченням, може знадобитися допомога професіонала.

2. Оскільки це бета–версія, софт може іноді видавати незначні помилки.

2.3 Висновок за розділом

Для реалізації цього проекту було обрано:

- Arduino NANO 1шт;
- I2C модуль;
- LCD дисплей 2004;
- дата кабель USB mini;
- датчик температури Ds18b20;
- комп’ютерний кулер;
- RGB стрічки;
- MOSFET транзисторів;
- резисторів;
- макетна плата;
- molex роз'єм.

Для створення прошивки програмного коду була обрана прграмма Arduino IDE 1.8.0 та утиліта Open Hardware Monitor.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРІНГУ

3.1 Проектування апаратного комплексу проекту

Збірку Монітору стану заліза ПК треба почати з того, ща треба припаяти ніжки до I2C модуля, а потім цей модуль припаяти до LCD дисплею (якщо він не припаєний).

Після цього потрібно підключити Arduino в до I2C проводами

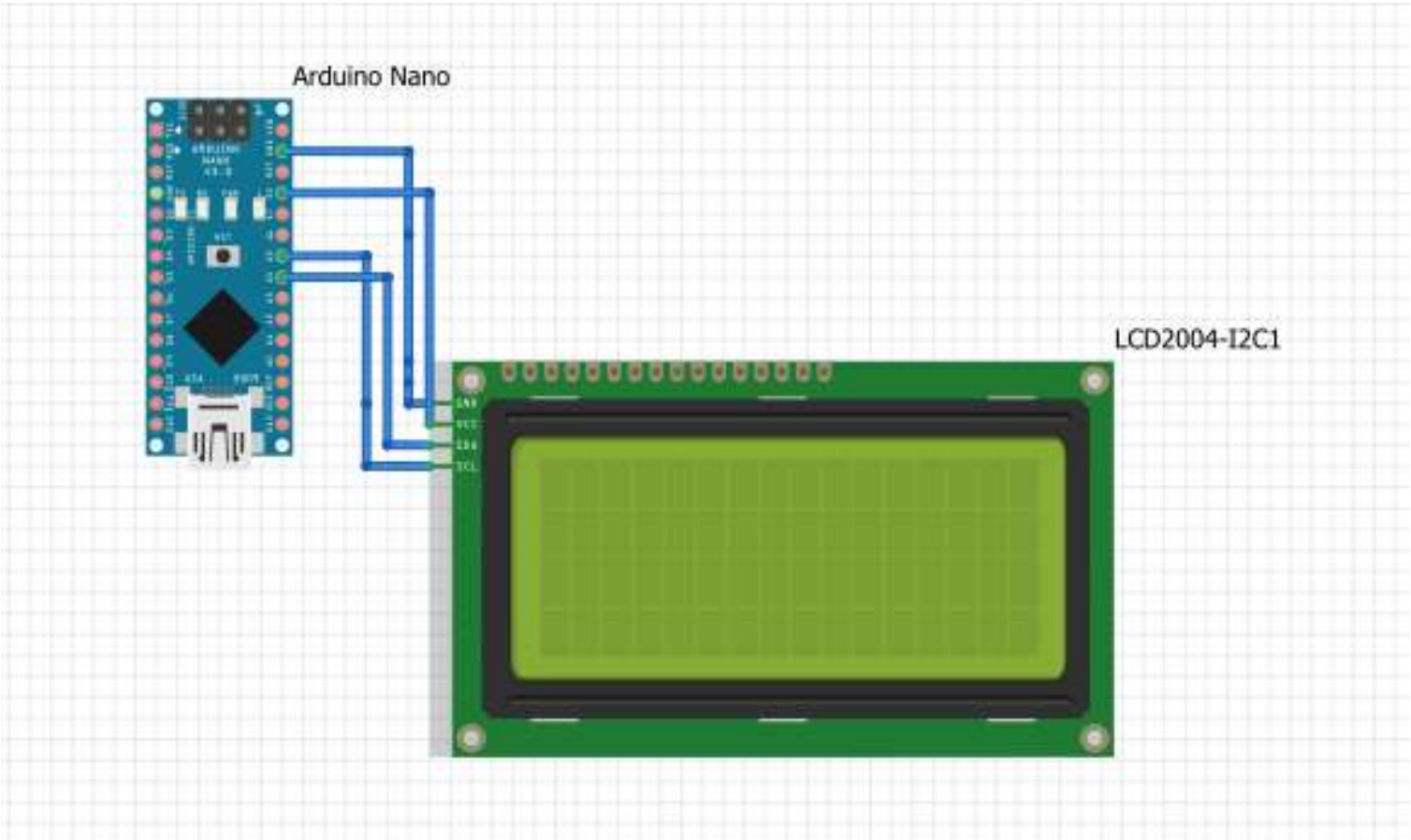


Рисунок 3.1 Підключення Arduino к I2C

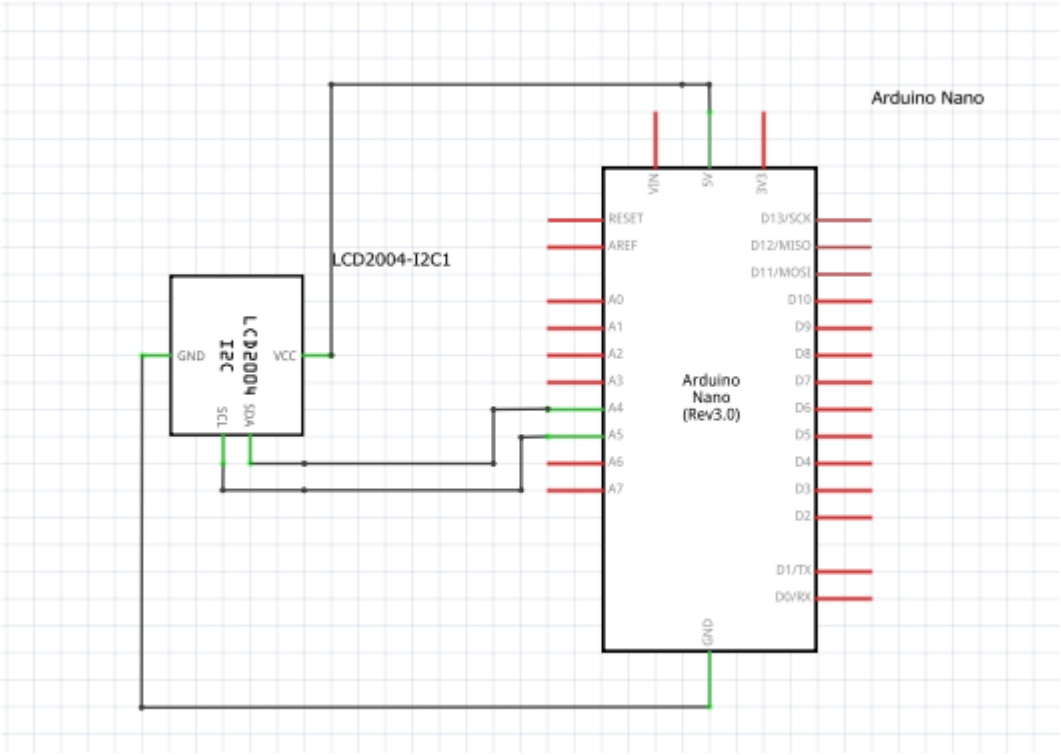


Рисунок 3.2 Ініціалізація схеми

Потім для живлення та прошивки треба підключити Arduino до комп’ютера кабелем USB micro.

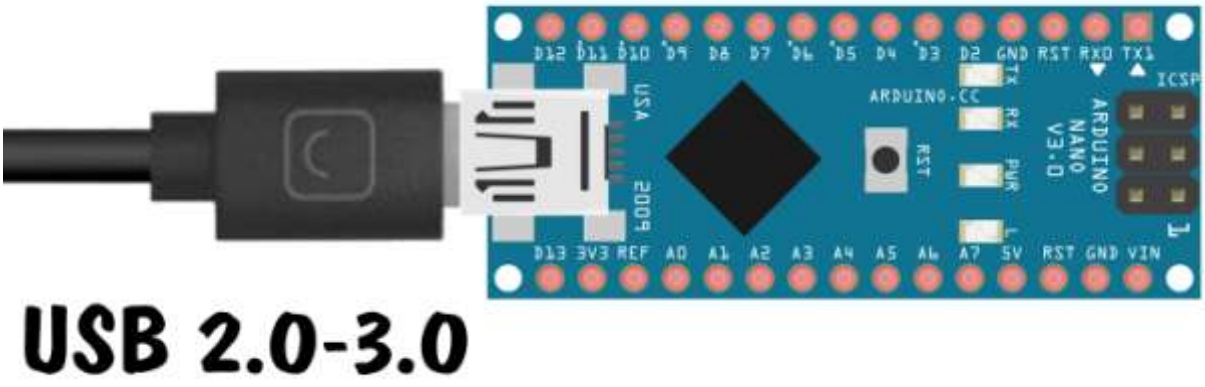


Рисунок 3.3 Подання живлення до Arduino

Після цього вирішили модифікувати проект. Було додано RGB стрічку, кулера для комп'ютера та датчик температури.

Перший MOSFET керує кулером за допомогою ШІМ сигналу. Три інших MOSFET управляють кольорами RGB стрічки. Кожен MOSFET відповідає за окремий колір так само ШИМ сигналом.

Складання компонентів було проведено на друкованій платі.

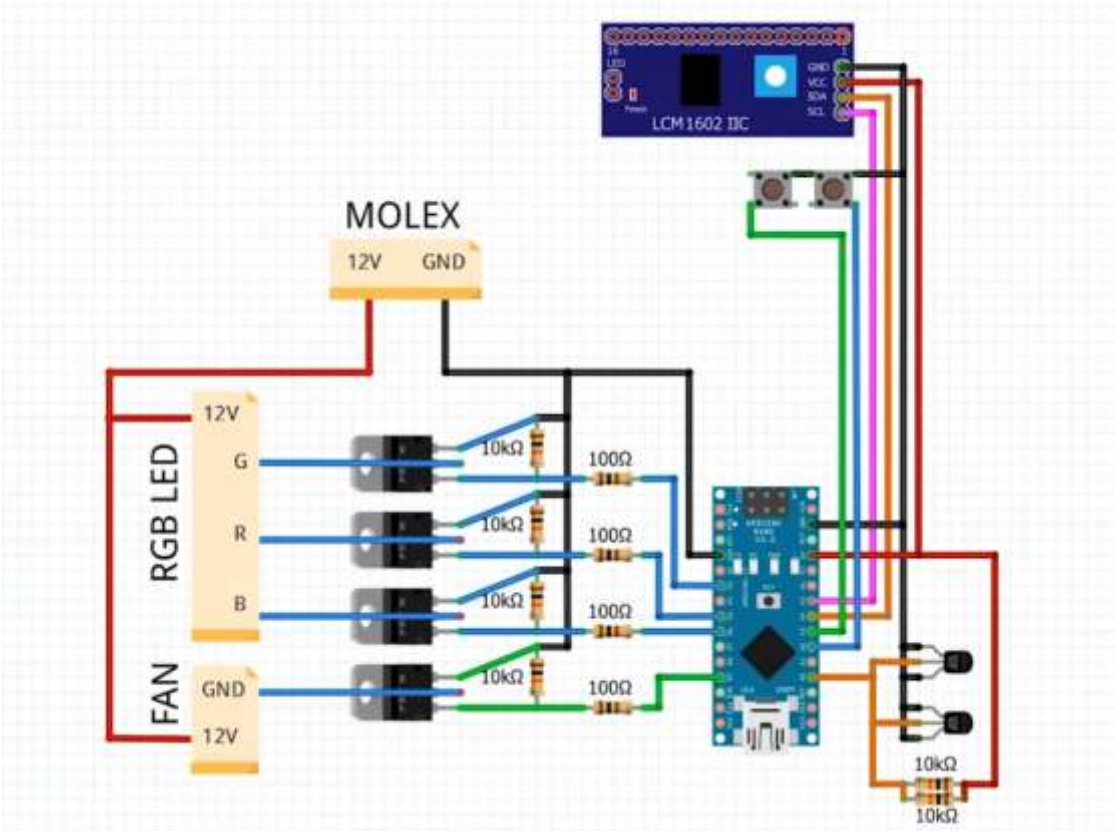
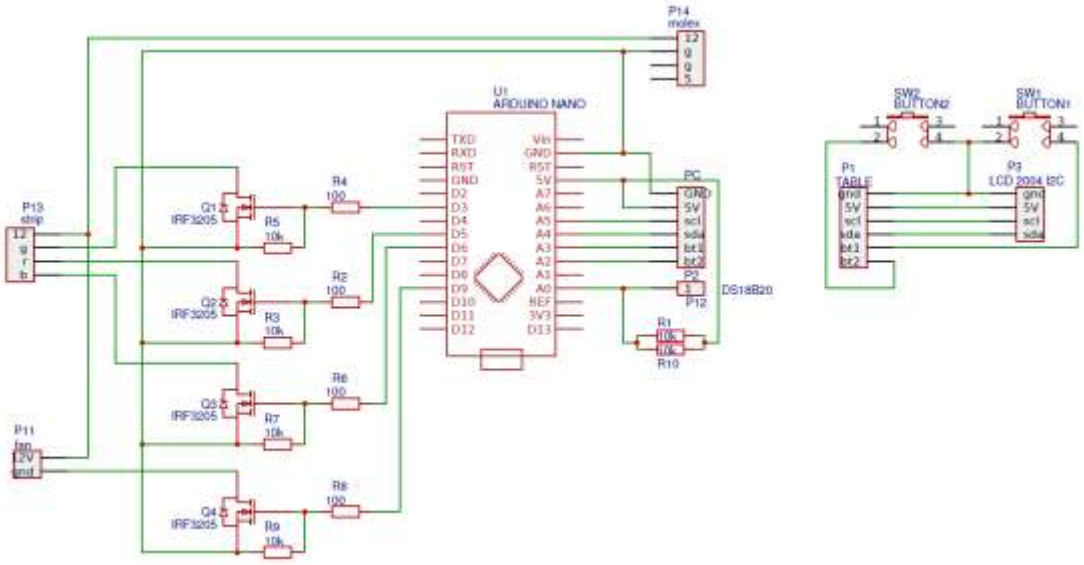


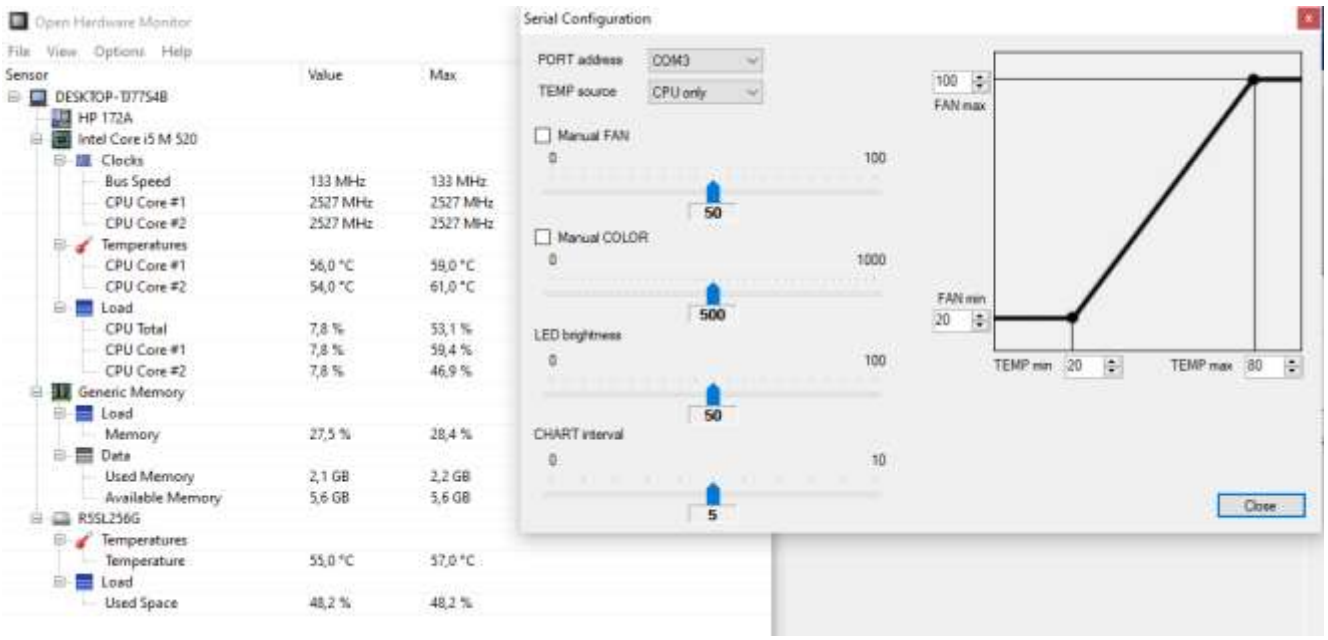
Рисунок 3.4 – Повна схема приладу



Рисунокунок 3.5 – Ініціалізація повної схеми

Після підключення Arduino до до кабелю живлення ми підключаемо кабель молекс до блоку живлення, для додаткового живлення.

Після підключення буде виводити інформацію на LCD екрані з утиліти



open hardware monitor.

Рисунок 3.6 – Інтерфейс утиліти open hardware monitor.

На першому екрані виводиться температура та завантаженість CPU, GPU, GPUmem, RAMuse. На другому екрані виводиться поточна швидкість вентилятора, температура датчика, температура материнської плати, жорсткого диска та час з моменту старту системи.

Далі потрібно зайти в утиліту open hardware monitor і вибрати порт.

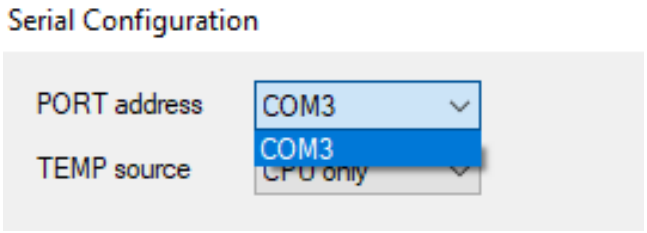


Рисунок 3.7 – Вибір порту

За замовчуванням швидкість вентилятора залежить від температури. Джерелом температури можна вибрати CPU, GPU, максимум між процесором та відеокартою, та датчиком.

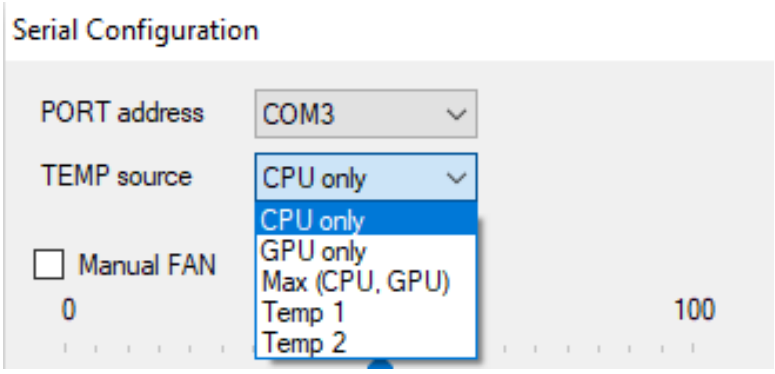


Рисунок 3.8 – Вибір джерела температури

Регулювання температури та швидкості відбувається за лінійним законом.

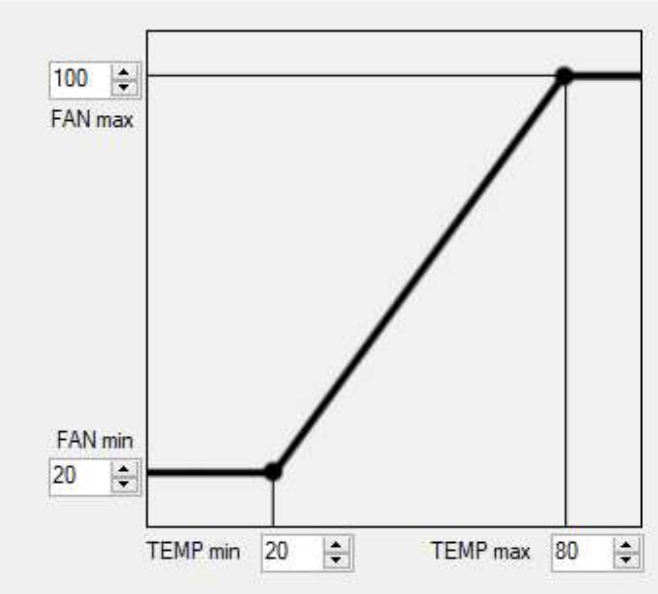


Рисунок 3.9 – Регулювання температури та швидкості

Також є функція ручного управління вентилятора і самостійно регулювати швидкість обертання кулера.

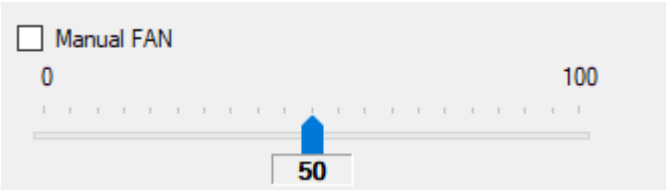


Рисунок 3.10 – Ручне управління кулером

Колір світлодіодної стрічки за промовчанням від вибраного джерела температури. На низьких температурах стрічка світиться синім кольором, потім при підвищенні температури вона світиться зеленим, жовтим і в самому кінці світиться червоним.

Також ручне керування світлом можна поставити і на світлодіодну стрічку і вибрати будь–який колір.

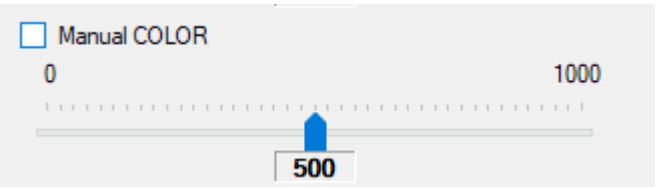


Рисунок 3.11 – Ручне керування світлом

Також нижче можна керувати яскравість світлодіодної стрічки.

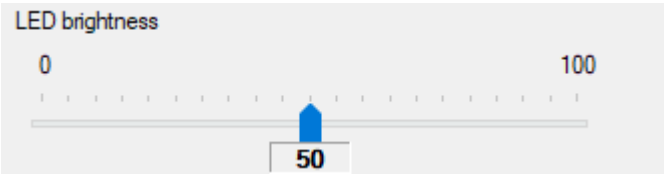


Рисунок 3.12 – Керування яскравістю

Ще нижче буде налаштування інтервалу оновлення графіків від 1 до 10 секунд.

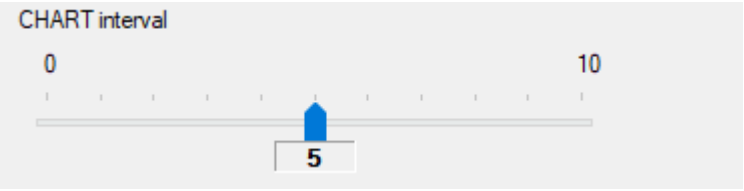


Рисунок 3.13 – Налаштування інтервалу оновлення графіків

3.2 Тестування та налагодження системи

Для живлення та роботоздатності проєкту потрібно підключити Arduino до комп’ютера та для додаткового живлення підключити Molex кабель.

Для запуску треба уввімкнути транслювання данних в утиліті open hardware monitor.

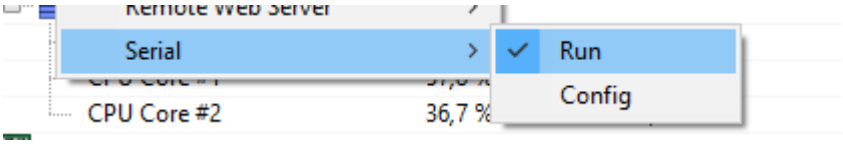


Рисунок 3.14 – Уввімкнення транслювання

Після підключення до живлення, компілювання Arduino та транслювання данних проєкт виглядае так.

Рисунок 3.15 Зібраний проект у дії.

3.3 Висновок за розділом

Використовуючи обрані у другому розділі компоненти, був створений Монітору стану заліза ПК. За допомогою підключаємого функціоналу створено програмний код прошивки плати. Опис диплому:

- Виведення основних параметрів заліза на зовнішній дисплей LCD;
    - Температура: CPU, GPU, материнська плата, найгарячіший HDD;
    - Рівень завантаження: CPU, GPU, RAM, відеопам'ять;
    - Графіки зміни вищезазначених параметрів за часом;
    - Температура зовнішніх датчиків (DS18B20);
    - Поточний рівень швидкості зовнішніх вентиляторів;
- Управління великою кількістю 12 вольтових 2, 3, 4 провідних вентиляторів;
    - Автоматичне керування швидкістю пропорційно температурі;
    - Ручне керування швидкістю з інтерфейсу програми;
- Управління RGB світлодіодною стрічкою;
    - Управління кольором у пропорційно температурі (синій – зелений – жовтий – червоний) ;
    - Ручне керування кольором із інтерфейсу програми;
    - Управління яскравістю.

ВИСНОВОКИ

Здійснивши огляд предметної області створення системы моніторингу апаратної частини ПК встановлено, що створення цього монітору є гарним рішенням для модифікування свого ПК, для спостереження за станом температури та для покращення своїх навичок в програмуванні. Ця інформація важлива для того, щоб дізнатися при перегріві комплектуючих.

Для реалізації цього проекту було обрано:

- Arduino NANO 1шт;
- I2C модуль;
- LCD дисплей 2004;
- Дата кабель USB mini;
- датчик температури Ds18b20;
- комп’ютерний кулер;
- RGB стрічки;
- MOSFET транзисторів;
- резисторів;
- макетна плата;
- molex роз'єм.

Для створення прошивки програмного коду була обрана прграмма Arduino IDE 1.8.0 та утиліта Open Hardware Monitor.

Використовуючи обрані у другому розділі компоненти, був створений Монітору стану заліза ПК. За допомогою підключаємого функціоналу створено програмний код прошивки плати.

ПЕРЕЛІК ПОСИЛАНЬ

1. Опис Arduino NANO. [Електронний ресурс] / Режим доступу www. URL: https://arduinomaster.ru/platy–arduino/plata–arduino–nano/– 23.05.2022

2. Огляд интерфейсного модуля, I2C. . [Електронний ресурс] / Режим доступу www. URL: https://robotchip.ru/obzor–interfeysnogo–modulya–i2c/– 23.05.2022

3. Підключення дисплея LCD 1602 до arduino по i2c. [Електронний ресурс] / Режим доступу www. URL: https://arduinomaster.ru/datchiki–arduino/lcd–i2c–arduino–displey–ekran/– 25.05.2022

4. Опис Arduino NANO . [Електронний ресурс] / Режим доступу www. URL: http://arduino.ru/Hardware/ArduinoBoardNano – 25.05.2022

5. Дисплей Arduino [Електронний ресурс] / Режим доступу www. URL: https://robo02.ru/2018/03/22/arduino–display–connect–a–text–screen/– 26.05.2022

6. Офіційний Arduino [Електронний ресурс] / Режим доступу www. URL: https://www.arduino.cc/ – 26.05.2022

7. Г. Шилдт C++: руководство для начинающих, 2–е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 672 с.: ил. – Парал. тит. англ. – 3 000 экз. – ISBN 5–8459–0840–X. – С. 349 –28.05.2022

8. МОП–структура [Електронний ресурс] / Режим доступу www. URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%9E%D0%9F–%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0 – 28.05.2022

9. Опис резистора [Електронний ресурс] / Режим доступу www. URL: http://surl.li/cdxjj – 28.05.2022

10. Опис датчика температури DS18B20 [Електронний ресурс] / Режим доступу www. URL: https://www.mini–tech.com.ua/datchik–temperatury–ds18b20–v–korpuse–s–kabelem – 28.05.2022

11. Опис печатної плати [Електронний ресурс] / Режим доступу www. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%87%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0 – 28.05.2022
12. Опис RGB стрічки [Електронний ресурс] / Режим доступу www. URL: https://5watt.ua/blog/stati/svetodiodnye–lenty–rgb–rgbw–i–rgbww – 28.05.2022

ДОДАТКИ

Додаток А. Вихідний код програми для Arduino

```
// ———————————— НАСТРОЙКИ ————————————
—

// настройки пределов скорости и температуры по умолчанию (на случай
отсутствия связи)
byte speedMIN = 10, speedMAX = 90, tempMIN = 30, tempMAX = 70;
#define DRIVER_VERSION 1     // 0 – маркировка драйвера кончается на
4AT, 1 – на 4T
#define COLOR_ALGORITM 0     // 0 или 1 – разные алгоритмы изменения
цвета (строка 222)
#define ERROR_DUTY 90       // скорость вентиляторов при потере связи
// ———————————— НАСТРОЙКИ ————————————
—

// ———————————— ПИНЫ ————————————
#define FAN_PIN 9          // на мосфет вентиляторов
#define R_PIN 5            // на мосфет ленты, красный
#define G_PIN 3            // на мосфет ленты, зелёный
#define B_PIN 6            // на мосфет ленты, синий
#define BTN1 A3            // первая кнопка
#define BTN2 A2            // вторая кнопка
#define SENSOR_PIN 14      // датчик температуры
int RECV_PIN = 11;         // пульт
// ———————————— ПИНЫ ————————————
```

```
// ———————————— БИБЛИОТЕКИ ————————————
#include <OneWire.h>           // библиотека протокола датчиков
#include <DallasTemperature.h>  // библиотека датчика
#include <string.h>            // библиотека расширенной работы со строками
#include <Wire.h>              // библиотека для соединения
#include <LiquidCrystal_I2C.h>  // библтотека дислея
//#include <TimerOne.h>          // библиотека таймера
#include <IRremote.h>          // библиотека пульта
#include <TimeLib.h>
// ———————————— БИБЛИОТЕКИ ————————————

// ———— АВТОВЫБОР ОПРЕДЕЛЕНИЯ ДИСПЛЕЯ————————
// Если кончается на 4T – это 0x27. Если на 4AT – 0x3f
#if (DRIVER_VERSION)
LiquidCrystal_I2C lcd(0x27, 16, 2);
#else
LiquidCrystal_I2C lcd(0x3f, 16, 2);
#endif
// ———— АВТОВЫБОР ОПРЕДЕЛЕНИЯ ДИСПЛЕЯ————————

#define printByte(args)  write(args);
#define TEMPERATURE_PRECISION 9
// настройка даьчтков
OneWire oneWire(SENSOR_PIN);
DallasTemperature sensors(&oneWire);
DeviceAddress Thermometer1, Thermometer2;
```

```
// стартовый логотип
byte logo0[8] = {0b00011, 0b00110, 0b01110, 0b11111, 0b11011, 0b11001, 0b00000, 0b00000};
byte logo1[8] = {0b10000, 0b00001, 0b00001, 0b00001, 0b00000, 0b10001, 0b11011, 0b11111};
byte logo2[8] = {0b11100, 0b11000, 0b10001, 0b11011, 0b11111, 0b11100, 0b00000, 0b00000};
byte logo3[8] = {0b00000, 0b00001, 0b00011, 0b00111, 0b01101, 0b00111, 0b00010, 0b00000};
byte logo4[8] = {0b11111, 0b11111, 0b11011, 0b10001, 0b00000, 0b00000, 0b00000, 0b00000};
byte logo5[8] = {0b00000, 0b10000, 0b11000, 0b11100, 0b11110, 0b11100, 0b01000, 0b00000};
// значок градуса!!!! lcd.write(223);
byte degree[8] = {0b11100, 0b10100, 0b11100, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000};
// правый край полосы загрузки
byte right_empty[8] = {0b11111, 0b00001, 0b00001, 0b00001, 0b00001, 0b00001, 0b00001, 0b11111};
// левый край полосы загрузки
byte left_empty[8] = {0b11111, 0b10000, 0b10000, 0b10000, 0b10000, 0b10000, 0b10000, 0b11111};
// центр полосы загрузки
byte center_empty[8] = {0b11111, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111};
// блоки для построения графиков
```

```
byte row8[8] = {0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte row7[8] = {0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte row6[8] = {0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte row5[8] = {0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111, 0b11111};
byte row4[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111, 0b11111};
byte row3[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111, 0b11111};
byte row2[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111, 0b11111};
byte row1[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b11111};

char inData[108];       // массив входных значений (СИМВОЛЫ)
int PCdata[26];         // массив численных значений показаний с компьютера

byte blocks, halfs;
byte index = 0;
int display_mode = 0;
String string_convert;
unsigned long timeout, blink_timer, plot_timer;
boolean lightState, reDraw_flag = 1, updateDisplay_flag, updateTemp_flag, timeOut_flag = 1;
```

```
int duty, LEDcolor;
int k, b, R, G, B, Rf, Gf, Bf;
byte mainTemp;
byte lines[] = {4, 5, 7, 6};
String perc;
unsigned long sec;
unsigned int mins, hrs;
byte temp1, temp2;
boolean btn1_sig, btn2_sig, btn1_flag, btn2_flag;
decode_results results;  //для пульта
IRrecv irrecv(RECV_PIN); //для пульта

tmElements_t tm;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();     // включить пульт
  //Timer1.initialize(40);          // поставить частоту ШИМ 25 кГц (40 микросекунд)
  pinMode(R_PIN, OUTPUT);
  pinMode(G_PIN, OUTPUT);
  pinMode(B_PIN, OUTPUT);
  digitalWrite(R_PIN, 0);
  digitalWrite(G_PIN, 0);
  digitalWrite(B_PIN, 0);
  pinMode(BTN1, INPUT_PULLUP);
  pinMode(BTN2, INPUT_PULLUP);
```

```
  sensors.begin();
  sensors.getAddress(Thermometer1, 0);
  sensors.getAddress(Thermometer2, 1);
  sensors.setResolution(Thermometer1, TEMPERATURE_PRECISION);
  sensors.setResolution(Thermometer2, TEMPERATURE_PRECISION);
  // инициализация дисплея
  lcd.init();
  lcd.backlight();
  lcd.clear();           // очистить дисплей
  show_logo();           // показать логотип
  delay(2000);
  lcd.clear();           // очистить дисплей

  //Timer1.pwm(FAN_PIN, 400);  // включить вентиляторы на 40%
  delay(2000);             // на 2 секунды
  lcd.clear();             // очистить дисплей
  PCdata[8] = speedMAX;
  PCdata[9] = speedMIN;
  PCdata[10] = tempMAX;
  PCdata[11] = tempMIN;

  setTime(1);  // устанавливаем начальное время
}
// 8–maxFAN, 9–minFAN, 10–maxTEMP, 11–minTEMP, 12–mnlFAN

// ———————————————— ОСНОВНОЙ ЦИКЛ ————————————————
```

```
void loop() {
  parsing();                  // парсим строки с компьютера
  getTemperature();           // получить значения с датчиков температуры
  dutyCalculate();            // посчитать скважность для вентиляторов
  //Timer1.pwm(FAN_PIN, duty * 10);     // управлять вентиляторами
  LEDcontrol();               // управлять цветом ленты
  buttonsTick();              // опрос кнопок и смена режимов
  irTick();                   // опрос пульта и смена режимов
  updateDisplay();            // обновить показания на дисплее
  timeoutTick();              // проверка таймаута
  if (tm.Hour <7) {
    lcd.noDisplay();
  lcd.noBacklight();
  }  else {
    lcd.display();
  lcd.backlight();
  }
}
// ———————————————————— ОСНОВНОЙ ЦИКЛ ————————————————————


void irTick() {
   if (irrecv.decode(&results)) {
    switch (results.value) {
      case 16736925: display_mode++;
                     reDraw_flag = 1;
                     if (display_mode > 3) display_mode = 0;
```

```
                break;
      case 16754775: display_mode—;
                reDraw_flag = 1;
                if (display_mode < 0) display_mode = 3;
                break;
    };
    irrecv.resume();
  }
}

void buttonsTick() {
  btn1_sig = !digitalRead(BTN1);
  btn2_sig = !digitalRead(BTN2);
  if (btn1_sig && !btn1_flag) {
    reDraw_flag = 1;
    if (display_mode == 0) display_mode = 1;
    else if (display_mode == 1) display_mode = 0;
    else display_mode = 0;
    btn1_flag = 1;
  }
  if (!btn1_sig && btn1_flag) {
    btn1_flag = 0;
  }
  if (btn2_sig && !btn2_flag) {
    reDraw_flag = 1;
    if (display_mode == 2) display_mode = 3;
    else if (display_mode == 3) display_mode = 2;
```

```
    else display_mode = 2;
    btn2_flag = 1;
  }
  if (!btn2_sig && btn2_flag) {
    btn2_flag = 0;
  }
}

void getTemperature() {
  if (updateTemp_flag) {
    sensors.requestTemperatures();
    temp1 = sensors.getTempC(Thermometer1);
    temp2 = sensors.getTempC(Thermometer2);
    updateTemp_flag = 0;
  }
}
void LEDcontrol() {
  b = PCdata[16];
  if (PCdata[13] == 1)          // если стоит галочка Manual Color
    LEDcolor = PCdata[15];      // цвет равен установленному ползунком
  else {                        // если нет
    LEDcolor = map(mainTemp, PCdata[11], PCdata[10], 0, 1000);
    LEDcolor = constrain(LEDcolor, 0, 1000);
  }

  if (COLOR_ALGORITM) {
    // алгоритм цвета 1
```

```
  // синий убавляется, зелёный прибавляется
  // зелёный убавляется, красный прибавляется
  if (LEDcolor <= 500) {
    k = map(LEDcolor, 0, 500, 0, 255);
    R = 0;
    G = k;
    B = 255 – k;
  }
  if (LEDcolor > 500) {
    k = map(LEDcolor, 500, 1000, 0, 255);
    R = k;
    G = 255 – k;
    B = 0;
  }

} else {
  // алгоритм цвета 2
  // синий максимум, плавно прибавляется зелёный
  // зелёный максимум, плавно убавляется синий
  // зелёный максимум, плавно прибавляется красный
  // красный максимум, плавно убавляется зелёный

  if (LEDcolor <= 250) {
    k = map(LEDcolor, 0, 250, 0, 255);
    R = 0;
    G = k;
    B = 255;
```

```
  }
  if (LEDcolor > 250 && LEDcolor <= 500) {
    k = map(LEDcolor, 250, 500, 0, 255);
    R = 0;
    G = 255;
    B = 255 – k;
  }
  if (LEDcolor > 500 && LEDcolor <= 750) {
    k = map(LEDcolor, 500, 750, 0, 255);
    R = k;
    G = 255;
    B = 0;
  }
  if (LEDcolor > 750 && LEDcolor <= 1000) {
    k = map(LEDcolor, 750, 1000, 0, 255);
    R = 255;
    G = 255 – k;
    B = 0;
  }
}

Rf = (b * R / 100);
Gf = (b * G / 100);
Bf = (b * B / 100);
analogWrite(R_PIN, Rf);
analogWrite(G_PIN, Gf);
analogWrite(B_PIN, Bf);
```

```
}

void dutyCalculate() {
  if (PCdata[12] == 1)              // если стоит галочка ManualFAN
    duty = PCdata[14];                    // скважность равна установленной ползунком
  else {                          // если нет
    switch (PCdata[18]) {
      case 0: mainTemp = PCdata[0];             // взять опорную температуру как CPU
        break;
      case 1: mainTemp = PCdata[1];             // взять опорную температуру как GPU
        break;
      case 2: mainTemp = max(PCdata[0], PCdata[1]);   // взять опорную температуру как максимум CPU и GPU
        break;
      case 3: mainTemp = temp1;
        break;
      case 4: mainTemp = temp2;
        break;
    }
    duty = map(mainTemp, PCdata[11], PCdata[10], PCdata[9], PCdata[8]);
    duty = constrain(duty, PCdata[9], PCdata[8]);
  }
  if (!timeOut_flag) duty = ERROR_DUTY;             // если пропало соединение, поставить вентиляторы на ERROR_DUTY
```

```
}
void parsing() {
  while (Serial.available() > 0) {
    char aChar = Serial.read();
    if (aChar != 'E') {
      inData[index] = aChar;
      index++;
      inData[index] = '\0';
    } else
    {
      char *p = inData;
      char *str;
      index = 0;
      String value = "";
      while ((str = strtok_r(p, ";", &p)) != NULL) {
        string_convert = str;
        PCdata[index] = string_convert.toInt();
        index++;
      }
      index = 0;
      updateDisplay_flag = 1;
      updateTemp_flag = 1;
      tm.Second = PCdata[21];
      tm.Hour = PCdata[19];
      tm.Minute = PCdata[20];
      tm.Day = PCdata[22];
      tm.Month = PCdata[23];
```

```
      tm.Year = PCdata[24] – 1970;
      setTime(makeTime(tm));
    }
    timeout = millis();
    timeOut_flag = 1;
  }
}

void updateDisplay() {
  if (updateDisplay_flag) {
    if (reDraw_flag) {
      lcd.clear();
      switch (display_mode) {
        case 0: draw_labels_11();
          break;
        case 1: draw_labels_12();
          break;
        case 2: draw_labels_21();
          break;
        case 3: draw_labels_22();
          break;
      }
      reDraw_flag = 0;
    }
    switch (display_mode) {
      case 0: draw_stats_11();
        break;
```

```
      case 1: draw_stats_12();
        break;
      case 2: draw_stats_21();
        break;
      case 3: draw_stats_22();
        break;
      case 50: debug();
        break;
    }
    updateDisplay_flag = 0;
  }
}

void draw_stats_11() {
  lcd.setCursor(4, 0); lcd.print(PCdata[0]); lcd.write(223);
  lcd.setCursor(13, 0); lcd.print(PCdata[4]);
  if (PCdata[4] < 10) perc = "% ";
  else if (PCdata[4] < 100) perc = "%";
  else perc = "";  lcd.print(perc);
  lcd.setCursor(4, 1); lcd.print(PCdata[1]); lcd.write(223);
  lcd.setCursor(13, 1); lcd.print(PCdata[5]);
  if (PCdata[5] < 10) perc = "% ";
  else if (PCdata[5] < 100) perc = "%";
  else perc = "";  lcd.print(perc);

  for (int i = 0; i < 2; i++) {
    byte line = ceil(PCdata[lines[i]] / 16);
```

```
    lcd.setCursor(7, i);
    if (line == 0) lcd.printByte(1)
      else lcd.printByte(4);
    for (int n = 1; n < 5; n++) {
      if (n < line) lcd.printByte(4);
      if (n >= line) lcd.printByte(2);
    }
    if (line == 6) lcd.printByte(4)
      else lcd.printByte(3);
  }
}

void draw_stats_12() {
  lcd.setCursor(13, 0); lcd.print(PCdata[7]);
  if (PCdata[7] < 10) perc = "% ";
  else if (PCdata[7] < 100) perc = "%";
  else perc = "";  lcd.print(perc);
  lcd.setCursor(13, 1); lcd.print(PCdata[6]);
  if (PCdata[6] < 10) perc = "% ";
  else if (PCdata[6] < 100) perc = "%";
  else perc = "";  lcd.print(perc);

  for (int i = 0; i < 2; i++) {
    byte line = ceil(PCdata[lines[i + 2]] / 16);
    lcd.setCursor(7, i);
    if (line == 0) lcd.printByte(1)
      else lcd.printByte(4);
```

```
    for (int n = 1; n < 5; n++) {
      if (n < line) lcd.printByte(4);
      if (n >= line) lcd.printByte(2);
    }
    if (line == 6) lcd.printByte(4)
      else lcd.printByte(3);
  }
}

void draw_stats_21() {
  lcd.setCursor(13, 0); lcd.print(duty);
  if ((duty) < 10) perc = "% ";
  else if ((duty) < 100) perc = "%";
  else perc = "";  lcd.print(perc);

  lcd.setCursor(3, 1); lcd.print(temp1); lcd.write(223);
  lcd.setCursor(11, 1); lcd.print(temp2); lcd.write(223);

  byte line = ceil(duty / 16);
  lcd.setCursor(6, 0);
  if (line == 0) lcd.printByte(1)
    else lcd.printByte(4);
  for (int n = 1; n < 5; n++) {
    if (n < line) lcd.printByte(4);
    if (n >= line) lcd.printByte(2);
  }
  if (line == 6) lcd.printByte(4)
```

```
      else lcd.printByte(3);
    }
    void draw_stats_22() {
      lcd.setCursor(2, 0); lcd.print(PCdata[2]); lcd.write(223);
      lcd.setCursor(10, 0); lcd.print(PCdata[3]); lcd.write(223);

      char lcd_time[10];
      snprintf(lcd_time, sizeof(lcd_time), "%02d:%02d",PCdata[19],PCdata[20]);
      char lcd_date[12];
      snprintf(lcd_date,                                    sizeof(lcd_date),
"%02d/%02d/%04d",PCdata[22],PCdata[23],PCdata[24]);

      lcd.setCursor(0, 1);
      lcd.print(lcd_time);

      lcd.setCursor(6, 1);
      lcd.print(lcd_date);
    }
    void draw_labels_11() {
      lcd.createChar(0, degree);
      lcd.createChar(1, left_empty);
      lcd.createChar(2, center_empty);
      lcd.createChar(3, right_empty);
      lcd.createChar(4, row8);
      lcd.setCursor(0, 0);
      lcd.print("CPU:");
      lcd.setCursor(0, 1);
```

```
  lcd.print("GPU:");
}
void draw_labels_12() {
  lcd.createChar(0, degree);
  lcd.createChar(1, left_empty);
  lcd.createChar(2, center_empty);
  lcd.createChar(3, right_empty);
  lcd.createChar(4, row8);
  lcd.setCursor(0, 0);
  lcd.print("GPUmem:");
  lcd.setCursor(0, 1);
  lcd.print("RAMuse:");
}
void draw_labels_21() {
  lcd.createChar(0, degree);
  lcd.createChar(1, left_empty);
  lcd.createChar(2, center_empty);
  lcd.createChar(3, right_empty);
  lcd.createChar(4, row8);

  lcd.setCursor(0, 0);
  lcd.print("FANsp:");
  lcd.setCursor(0, 1);
  lcd.print("T1: ");
  lcd.setCursor(8, 1);
  lcd.print("T2:");
}
```

```
void draw_labels_22() {
  lcd.createChar(0, degree);
  lcd.createChar(1, left_empty);
  lcd.createChar(2, center_empty);
  lcd.createChar(3, right_empty);
  lcd.createChar(4, row8);

  lcd.setCursor(0, 0);
  lcd.print("M:");
  lcd.setCursor(6, 0);
  lcd.print("HDD:");
}



void timeoutTick() {
  if (millis() – timeout > 5000) {
    if (timeOut_flag == 1) {
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("DISCONNECTED");
      timeOut_flag = 0;
      reDraw_flag = 1;
    }
    char lcd_time_i[10];
    snprintf(lcd_time_i, sizeof(lcd_time_i), "%02d:%02d",hour(),minute());
    char lcd_date_i[12];
```

```
            snprintf(lcd_date_i,                                    sizeof(lcd_date_i),
"%02d/%02d/%04d",day(),month(),year());
            lcd.setCursor(0, 1);
            lcd.print(lcd_time_i);
            lcd.setCursor(6, 1);
            lcd.print(lcd_date_i);
          }
        }
        void show_logo() {
          lcd.createChar(0, logo0);
          lcd.createChar(1, logo1);
          lcd.createChar(2, logo2);
          lcd.createChar(3, logo3);
          lcd.createChar(4, logo4);
          lcd.createChar(5, logo5);
          lcd.setCursor(0, 0);
          lcd.printByte(0);
          lcd.printByte(1);
          lcd.printByte(2);
          lcd.setCursor(0, 1);
          lcd.printByte(3);
          lcd.printByte(4);
          lcd.printByte(5);
          lcd.setCursor(9, 0);
          lcd.print("PC");
          lcd.setCursor(5, 1);
          lcd.print("MONITORING");
```

```
}

void debug() {
  lcd.clear();
  lcd.setCursor(0, 0);
  for (int j = 0; j < 5; j++) {
    lcd.print(PCdata[j]); lcd.print("  ");
  }
  lcd.setCursor(0, 1);
  for (int j = 6; j < 10; j++) {
    lcd.print(PCdata[j]); lcd.print("  ");
  }
  lcd.setCursor(0, 2);
  for (int j = 10; j < 15; j++) {
    lcd.print(PCdata[j]); lcd.print("  ");
  }
  lcd.setCursor(0, 3);
  for (int j = 15; j < 18; j++) {
    lcd.print(PCdata[j]); lcd.print("  ");
  }
}
```