МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ ЦИФРОВОГО ГОДИННИКА-ТЕРМОМЕТРА НА ПЛАТФОРМІ ARDUINO

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Дениса ПИРОЖЕНКО

Науковий керівник:
PhD, кандидат технічних наук
Андрій-Володимир МІДИК

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім'я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2025 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____Ігор ФЕДЧАК_
(підпис)

Львів
2025

СПИСОК УМОВНИХ СКОРОЧЕНЬ

ПК	персональний комп’ютер
МК	мікроконтролер
ЕК	електрична схема
АЗ	апаратне забезпечення
ПЗ	програмне забезпечення
ЦП	цифровий пристрій
САПР	система автоматизованого проектування
Arduino Micro	апаратно-програмна платформа на МК ATmega32u4
AVR	сімейство 8 - розрядних мікроконтролерів фірми Atmel
Arduino IDE	інтегроване середовище розробки, яке використовується для написання і прошивки програмного коду в мікроконтролер плати Arduino
FLASH	різновид напівпровідникової технології пам’яті, яка електричним методом перепрограмовується
EEPROM	(англ. Electrically Erasable Programmable Read-Only Memory) – постійна пам’ять, яка стирається і перепрограмовується електричним методом
ПЗП (ROM)	(англ. read-only memory) постійний запам’ятовуючий пристрій. Незалежна пам’ять, використовується для зберігання масиву незмінних даних.
ОЗП (RAM)	оперативна пам’ять або оперативний запам’ятовуючий пристрій (англ. Random Access Memory, RAM)
АЦП (ADC)	аналого-цифровий перетворювач
SPI	(англ. Serial Peripheral Interface, SPI bus - послідовний периферійний інтерфейс, шина SPI) - послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для забезпечення простого і

	недорогого високошвидкісного сполучення мікроконтролерів і периферії
I^2C	послідовна шина даних для зв’язку інтегральних схем, що використовує дві двонаправлені лінії зв’язку (SDA і SCL).

АНОТАЦІЯ

Пироженко Д., Мідик А.-В. (керівник). Розроблення цифрового годинника-термометра на платформі Arduino. Бакалаврська кваліфікаційна робота. –Львівський державний університет внутрішніх справ, Львів, 2025.

У дипломній роботі створено апаратне та програмне забезпечення цифрового пристрою, що має функції годинника та термометра. Цифровий пристрій розроблено на платформі Arduino Micro з МК AVR ATmega32u4, до якої підключено мікросхему годинника реального часу DS1307, прецизійний давач температури LM35DZ та матричний світлодіодний дисплейний модуль 8x32 FC-16 з мікросхемами управління MAX7219 (дисплейний модуль з 4 точкових світлодіодних матриць 8x8 і мікросхем управління MAX7219). Мікроконтролер зчитує час і дату з мікросхеми RTC DS1307, температуру навколишнього середовища з давача LM35DZ та відображає (виводить) їх на матричному світлодіодному дисплеї. Пристрій має можливість налаштування поточного часу, дати, параметрів виводу інформації.

В роботі розроблено електричну принципову схему та модель цифрового годинника - термометра засобами САПР Proteus. Розроблено програмно-алгоритмічне забезпечення цифрового годинника – термометра в середовищі Arduino IDE. Проведено моделювання в емуляторі Proteus ISIS та дослідження макету цифрового годинника-термометра.

Ключові слова: цифровий пристрій, платформа Arduino Micro, мікроконтролер AVR ATmega32u4, мікросхема RTC DS1307, прецизійний давач температури LM35DZ, матричний світлодіодний дисплейний модуль 8x32 FC-16, мікросхема – драйвер MAX7219, САПР Proteus VSM, мова програмування С для вбудованих систем, вбудоване програмне забезпечення, середовище розробки програмного забезпечення для платформи Arduino – Arduino IDE

ABSTRACT

Pyrozhenko D., Midyk A.-V. (supervisor). Development of digital clock-thermometer based on the Arduino platform. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2025.

In the diploma thesis, hardware and software of the digital device with clock and thermometer functions have been created. It has functions of the digital clock and thermometer. The digital device is developed using the Arduino Micro platform based on AVR ATmega32u4 microcontroller. to which RTC DS1307, LM35DZ precise temperature sensor and LED matrix display module 8x32 FC-16 with the MAX7219 IC driver are connected.

The microcontroller reads time and date from the real-time clock DS1307, temperature from the LM35DZ sensor and output them to the LED matrix display. The digital device allows a user to set the current date and time, alarm time, as well information output parameters.

In the work, the electric circuit and model of the digital clock - thermometer are developed using the CAD system - Proteus VSM. The device operation algorithm and software for the digital clock-thermometer are developed in C programming language using Arduino IDE. The simulation of the digital device is performed in Proteus ISIS. The prototype of the digital clock – thermometer is developed and tested.

Keywords: digital device, Arduino Micro, AVR ATmega32u4 microcontroller, real-time clock DS1307, precise temperature sensor LM35DZ, 8x32 LED matrix display module FC-16 (4x8x8 (8x32) MAX7219 dot matrix LED display module), MAX7219 IC driver, CAD Proteus VSM, embedded C, embedded software, software development tool Arduino IDE.

ЗМІСТ

ВСТУП

Розвиток мікроелектроніки привів до появи мікроконтролерів, різних давачів та периферійних модулів, що дає змогу створювати недорогі цифрові пристрої. Зокрема, таким є цифровий годинник - термометр на платформі Arduino.

На сьогодні мікроконтролери (МК) можна зустріти в багатьох сучасних пристроях і приладах. Вонии широко використовуються в різних цифрових пристроях, а саме в побутовій техніці, автомобілях, телевізорах, промислових роботах тощо. МК представляє собою мікросхему, яка призначена для управління електронними пристроями. Типовий МК об'єднює на одному кристалі функції процесора та периферійних пристроїв, містить ОЗП і ПЗП. По суті, це однокристальний комп'ютер, який здатний виконувати відносно прості задачі. На відміну від звичайних комп'ютерних мікропроцесорів, в МК часто використовується гарвардська архітектура пам'яті, тобто розділене зберігання даних в ОЗП, а команд в ПЗП. МК зазвичай також має вбудовану енергонезалежну пам'ять для зберігання програми і даних. В сучасних мікроконтролерах в одній мікросхемі є достатньо потужний процесор та різна периферія, що дозволяє суттєво зменшити розміри, енергоспоживання і вартість розроблених на ньому пристроїв. Програмування мікроконтролерів, як правило, здійснюють на мовах Assembler, C/C++, хоча є компілятори і для інших мов. Популярними серед розробників є 8 - розрядні мікроконтролери PIC фірми Microchip Technology, AVR фірми Atmel (з 2016 року Microchip), 16 - розрядні MSP430 від Texas Instruments, а також 32 - розрядні МК архітектури ARM STM32 фірми STMicroelectronics.

Аналіз останніх досліджень і публікацій. Основи досліджень можливостей використання мікроконтролерів для створення багатофункціональних пристроїв були закладені працями таких учених та інженерів Чарльз Страк, Філіп Фрайс, Франко Малоберті. Зокрема, Філіп

Фрайс та інженери компанії Atmel – розробники мікроконтролерів AVR, на яких базується платформа Arduino, що зробило мікроконтролери доступними для широкого кола розробників та інженерів. Франко Малоберті зробив значний внесок у розвиток аналогово-цифрових схем і сенсорних технологій, що є критичними для сучасних багатофункціональних мікроконтролерних систем. Ці праці та відкриття стали основою для подальших досліджень у сфері вбудованих систем, зокрема для створення цифрових годинників-термометрів та інших інтелектуальних пристроїв на базі мікроконтролерів. Методи та засоби використання мікроконтролерів для створення багатофункціональних пристроїв постійно розвиваються та вдосконалюються. Тому подальше вдосконалення цифрового годинника-термометра з додатковими функціями, такими як вимірювання температури та атмосферного тиску, залишається актуальним напрямом досліджень.

Метою роботи є розробка апаратного і вбудованого програмного забезпечення цифрового пристрою з функціями годинника і термометра.

Для досягнення мети необхідно вирішити наступні **завдання**:

- розробити цифровий годинник-термометр на основі наступних даних: плата Arduino Micro на 8 - розрядному МК AVR ATmega32u4; прецизійний давач температури за Цельсієм LM35DZ з діапазоном вимірювання: -55...+150°C, з точністю: ±0,5°C; мікросхема годинника реального часу RTC DS1307; матричний світлодіодний дисплей FC-16 8x32 з мікросхемами драйверів MAX7219; 4 двохконтактні кнопки на 4 виводи 6×6×5 мм;
- спроєктувати багатофункціональний цифровий годинник- термометр засобами САПР ProteusVSM;
- розробити алгоритм роботи цифрового годинника-термометра;
- створити програмний модуль роботи з годинником реального часу;
- створити програмне забезпечення для роботи з давачем температури;
- створити програмний модуль виводу інформації на дисплейний модуль з світлодіодних матриць;

- розробити основне програмне забезпечення багатофункціонального цифрового годинника-термометра згідно алгоритму його роботи;
- провести моделювання роботи спроєктованого багатофункціонального цифрового годинника-термометра та проаналізувати отримані результати в емуляторі Proteus ISIS;
- створити і дослідити макет цифрового годинника-термометра.

Об'єкт дослідження – проєктування цифрового пристрою, що містить годинник-термометр, давач температури LM35DZ, матричний світлодіодний дисплей FC-16 8x32 з мікросхемами драйверів MAX7219.

Предмет дослідження – моделі та засоби проєктування цифрового пристрою, що містить годинник-термометр, давач температури LM35DZ, матричний світлодіодний дисплей FC-16 8x32 з мікросхемами драйверів MAX7219, на основі МК ATmega32u4.

Методи досліджень. Під час виконання роботи застосовано низку загальновідомих методів пізнання, які дозволили у повному обсязі досягти мети та вирішити завдання дослідження. Так, за допомогою бібліометричного методу здійснено опрацювання опублікованих наукових праць, що дозволило осягнути динаміку розвитку даної проблеми. За допомогою різних філософських (діалектичного, феноменологічного, синергетичного, аксіологічного) методів пізнання здійснено всебічний аналіз поглядів на предмет дослідження. Використання значної кількості загальнонаукових методів (структурного, індуктивного, дедуктивного, аналізу, синтезу, моделювання) дозволило досягнути виконання визначених вище теоретичних та практичних завдань дослідження.

Структура роботи. Кваліфікаційна робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Обсяг основного тексту роботи складає 86 сторінок, 55 рисунків, 17 таблиць, 3 додатки і 12 бібліографічних джерел. Загальний обсяг роботи – 108 сторінок.

РОЗДІЛ І
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Вимірювальні величини иа їх представлення

В світі прийнято два формати представлення часу: 24 – годинний і 12 - годинний. В дванадцятигодинному форматі визначення часу 24 години доби розбивають на два інтервали (відрізки) по 12 годин, які позначають AM/a.m. (з лат. ante meridiem) – до полудня, і PM/p.m. (з лат. post meridiem) – після полудня. Дванадцяти годинний формат представлення часу переважає в Австралії. Канаді (за виключенням Квебека), США, Нова Зеландія і Філіпіни. Спільно з загальноприйнятим 24 – годинним форматом, 12 – годинний формат використовується в Албанії, Великобританії, Бразилії та в деяких інших алгломовних країнах, в Греції, Ірландії, канадському Квебеку, Туреччині і Франції. В них позначають час будь-якими способами. В решти країн прийнято двадцятичотирьохгодинний формат часу.

Дванадцятигодинний формат широко використовують у неформальному спілкуванні. Нариклад, замість a.m. та p.m. можна почути такі описові конструкції як “ранку”, “дня”, “вечора”, “ночі”, “полудня” та інші. Дванадцятигодинний формат є поширеним у неформальному спілкуванні завдяки коротким числівникам, а також дванадцятигодинним циклом годинника з класичним циферблатом. Недивлячись на міжнародні стандарти позначення 12 – годинного формату часу (ISO 8601) – досі немає однозначності між позначенням такого часового проміжку як “полудень” та “північ”. Деякі вказують полудень як “12 a.m.” (“12 ante meridiem”, або “12 година до середини дня”). За цією логікою північ також можна позначити як “12 p.m.” (“12 post meridiem або 12 година після попередньої середини дня”). Національний морський музей Грінвіча пропонує позначати північ як “12 годин ночі”, а полудень – як “12 година дня” [1].

Таблиця 1.1.

Порівняльна таблиця 24 – годинного і 12-голинного форматів часу

24-годинний формат	12-годинний формат	Промова
00:00 (опівночі)	12:00 a.m. (опівночі)	дванадцята (година) ночі опівночі
01:00	1:00 a.m.	перша (година) ночі
02:00	2:00 a.m.	друга (година) ночі
03:00	3:00 a.m.	третя (година) ночі
04:00	4:00 a.m.	четверта (година) ранку
05:00	5:00 a.m.	п’ята (година) ранку
06:00	6:00 a.m.	шоста (година) ранку
07:00	7:00 a.m.	сьома (година) ранку
08:00	8:00 a.m.	восьма (година) ранку
09:00	9:00 a.m.	дев’ята (година) ранку
10:00	10:00 a.m.	десята (година) ранку
11:00	11:00 a.m.	одинадцята (година) ранку
12:00 (опівдні)	12:00 p.m. (опівдні)	дванадцята (година) дня
13:00	1:00 p.m.	перша (година) дня
14:00	2:00 p.m.	друга (година) дня
15:00	3:00 p.m.	третя (година) дня
16:00	4:00 p.m.	четверта (година) дня
17:00	5:00 p.m.	п’ята (година) дня
18:00	6:00 p.m.	шоста (година) вечора
19:00	7:00 p.m.	сьома (година) вечора
20:00	8:00 p.m.	восьма (година) вечора
21:00	9:00 p.m.	дев’ята (година) вечора
22:00	10:00 p.m.	десята (година) вечора
23:00	11:00 p.m.	одинадцята (година) вечора
00:00 (опівночі)	12:00 a.m. (опівночі)	дванадцята година ночі (опівночі)

Багато американських сертифікованих посібників по стилю пропонують позначати північ в форматі 11.59 p.m., щоб підкреслити кінець однієї доби, а вже початок наступного дня позначати як 12.01 a.m.. Ця рекомендація часто застосовується для розкладів руху транспорту та юридичних угод США. До речі, саме через складнощі в розумінні та визначенні часу за 12 – годиниим форматом, армія США ще з часів Другої світової війни використовує 24 – годинний формат, який дозволяє уникнути помилок в навігації та позначенні часу воєнних дій. У випадку сумніву в тому, як правильно вказати час доби – можна відмовитися від цифр і сказати повним словом: midday – полудень, midnight – північ. В таблиці 2.1 наведено основні формати дати і часу, які використовуються в різних країнах.

Таблиця 1.2

Формати дати і часу в різних країнах

Країна/мова	Формат дати	Формат часу	Приклад дати	Приклад часу
Україна	DD.MM.YYYY	HH:MI:SS	20.02.2023	12:14:00
США	MM-DD-YYYY	HH:MI:SS	02-20-2023	12:14:00 (12:14:00 p.m.)
Міжнародний англійський	DD-MM-YYYY	HH:MI:SS	20-02-2023	12:14:00
Велико-британія	DD/MM/YYYY	HH:MI:SS	20/02/2023	12:14:00 (12:14:00 p.m.)
Німеччина	DD.MM.YYYY	HH:MI:SS	20.02.2023	12:14:00
Бельгія	DD/MM/YYYY	HH:MI:SS	20/02/2023	12:14:00
Бразилія	DD/MM/YYYY	HH:MI:SS	20/02/2023	12:14:00
Угорщина	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00
Данія	DD-MM-YYYY	HH.MI.SS	20-02-2023	12.14.00
Італія	DD/MM/YYYY	HH.MI.SS	20/02/2023	12.14.00
Іспанія	DD/MM/YYYY	HH:MI:SS	20/02/2023	12:14:00
Канада (франц.)	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00
Латинська Америка	DD/MM/YYYY	HH:MI:SS	20/02/2023	12:14:00
Голандія	DD-MM-YYYY	HH:MI:SS	20-02-2023	12:14:00
Норвегія	DD.MM.YYYY	HH:MI:SS	20.02.2023	12:14:00
Польща	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00

Португалія	DD-MM-YYYY	HH:MI:SS	20-02-2023	12:14:00
Сербія	DD.MM.YYYY	HH.MI.SS	20.02.2023	12.14.00
Словакія	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00
Словенія	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00
Фінляндія	YYYY-MM-DD	HH.MI.SS	2023-02-20	12.14.00
Франція	DD.MM.YYYY	HH:MI:SS	20.02.2023	12:14:00
Чехія	YYYY-MM-DD	HH:MI:SS	2023-02-20	12:14:00
Швейцаря	DD.MM.YYYY	HH,MI,SS	20.02.2023	12,14,00
Швеція	YYYY-MM-DD	HH.MI.SS	2023-02-20	12.14.00

Температура повітря – це велична, яка відображає степінь нагрітості повітря. Температура повітря є одним з самих мінливих показнків стану повітря. ЇЇ вимірюють з допомогою термометра, різних термодавачів. Існують різні шкали вимірювання температури: Цельсія (°C), Кельвіна (K), Фаренгейта (°F). По даних спостережень за певний період обчислюють середні (за добу, тиждень, місяць, рік) максимальні і мінімальні температури, а також амплітуду температур. У побуті найбільш поширеною є шкала Цельсія, в якій за 0 приймають точку замерзання води, а за 100° точку кипіння води при атмосферному тиску. Оскільки температура замерзання і кипіння води недостатньо добре визначена, на теперішній час шкалу Цельсія визначають через шкалу Кельвіна: градус Цельсія рівний Кельвіну, абсолютний нуль приймається за -273,15 °C. Шкала Цельсія практично дуже зручна, оскільки вода дуже поширена на нашій планеті, і на ній основане наше життя. Нуль Цельсія – особлива точка для метеорології, оскільки замерзання атмосферної води суттєво все змінює [10].

В Англії і США використовується шкала Фаренгейта, яку запропонував в 1724 році Г. Фаренгейт. В цій шкалі на 100 градусів розділений інтервал від температури самої холодної зими в місті, де проживав Фаренгейт, до температури людського тіла. Нуль градусів Цельсія – це 32 градуси Фаренгейта, а градус Фаренгейта рівний 5/9 градуса Цельсія. В даний час прийнято наступне визначення шкали Фаренгейта: це температурна шкала, 1 градус якої (1 °F) рівний 1/180 різниці температур кипіння води і танення льоду при атмосферному тиску, а точка танення льоду має температуру +32 °F.

Температура по шкалі Фаренгейта зв’язана з температурою по шкалі Цельсія (t °C) співвідношенням t °C = 5/9 (t °F - 32), тобто зміна температури на 1 °F відповідає зміні на 5/9 °C.

1.2. Огляд існуючих розробок цифрових годинників-термометрів

Провівши огляд літературних джерел та інтернет-ресурсів було отримано інформацію щодо розробок різноманітних цифрових пристроїв, що мають функції годинника і термометра [10, 12]. На сьогоднішній день пропонуються різні рішення щодо створення цифрових пристроїв, які побудовані на мікроконтролерах і мають засоби бездротового зв’язку. На рисунках 1.1 – 1.6 зображено різні розробки цифрових пристроїв з функціями годинника і термометра з виводом інформації на дисплей.

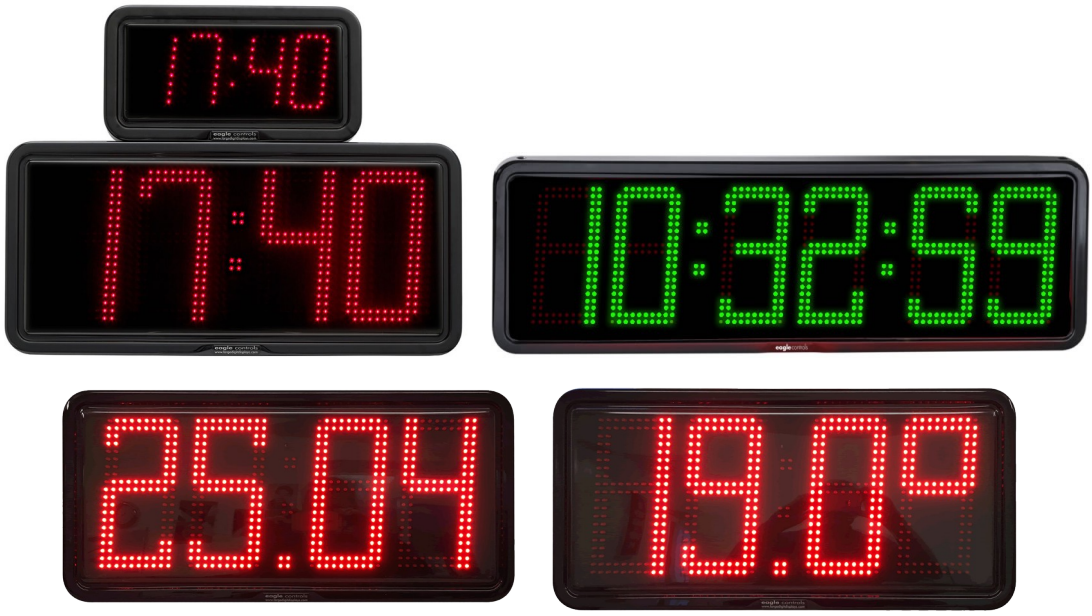


Рис.1.1. LED цифровий годинник термометр фірми eagle controls. Формати часу HRS:MINS або HRS:MINS:SECS. Діапазон вимірювання температури (прграмований): -50 ~ 99,9 °C або -58 ~ 212 °F. Формат дати: DAY/MONTH/YEAR

Рис.1.2. Цифровий термометр і годинник НТС-1

Таблиця 1.3

Характеристики багатофункціонального цифрового годинника з термометром і гігрометром

Особливості	вивід інформації на LCD, має функції термометра і гігрометра, можливість запису вимірювань температури.
Технічні характеристики	Діапазон вимірювання температури: -50°C до 70°C (-58 °F до 158 °F). Точність ±1. Діапазон вимірювання вологості:10% до 99% RH; Точність гігрометрп: ±5% RH Вибір одиниць вимірювання температури: °C або °F. Автоматичний запис MAX/MIN температури і вологості.
Живлення	1 x AAA 1,5V батарейка

Рис.1.3. Цифровий пристрій з функціями годинника і термометра

Цифровий пристрій з функціями годинника і термометра, виводить на LCD час, дату і день, підтримує 12/24 – годинні формати, має функцію smart

нічного світла для виводу часу і температури в темному приміщенні, може відоабражати темпреатуру в C/F, маю функцію будильника зі звуковим оповіщенням протягом 60 секунд і яка переактивовується знову через 8 хвилин, живитьеся від 3 х AAA батарейок [12].

Рис.1.4. USB LED годинник з термометром

USB LED годинник з термометром має 5 рівнів налаштування яскравості, відбражає час, дату і температуру, 2 режими будильника, на вибір шкала °C/°F, 12/24-годинний формат часу [12].

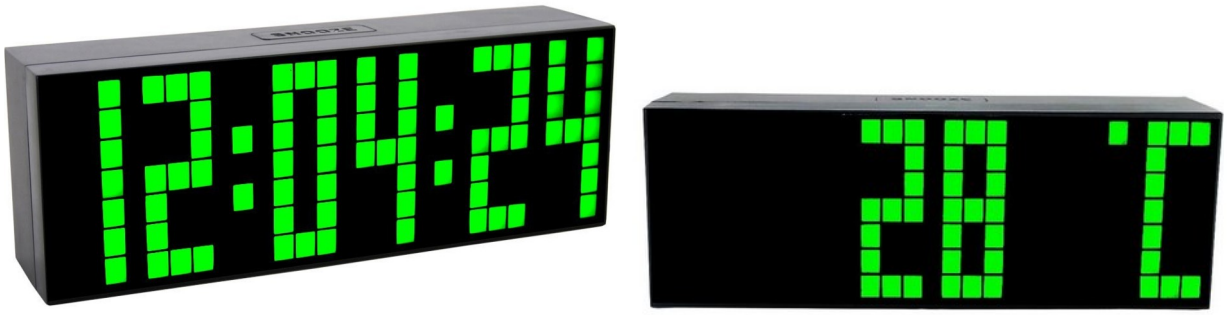


Рис.1.5. Цифровий настільний LED годинник з термометром

Рис.1.6. LED цифровий годинник з термометром 5”x4 BIGLED-4D5 для використання надворі

РОЗДІЛ ІІ
ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ГОДИННИКА-ТЕРМОМЕТРА

2.1. Система автоматизованого проектування і моделювання програмованих пристроїв Proteus VSM

Proteus VSM представляє собою потужну систему схемотехнічного проектування і моделювання, що базується на віртуальних моделях електронних елементів (рисунок 2.1). Важливою і корисною особливістю Proteus є змога проводити моделювання роботи програмованих пристроїв, які побудовані на мікропроцесорах, мікроконтролерах, DSP. Система Proteus складається з двох основних програм ISIS та ARES.

1) ISIS – це програма для редагування принципових електричних схем. Вона призначена для створення проектів пристроїв (за принциповою електричною схемою) з подальшою симуляцією їх роботи та передачі в ARES для розробки друкованих плат. Після відладки пристрою можна в ARES розвести для нього друковану плату. В ARES є підтримка автоматичного розміщення та трасування за існуючою схемою.
2) ARES – це програма для розробки друкованих плат, яка має вбудований автотрасувальник ELECTRA, автоматичне розміщенням компонентів на друкованій платі і хороший менеджер бібліотек.

В Proteus ISIS можна змоделювати роботу таких мікроконтролерів як: 8051, PIC, AVR, ARM7, Motorola, Basic Stamp. У внутрішній бібліотеці компонентів міститься різна довідкова інформація. Бібліотека підтримує мікроконтролери: 8051, PIC, HC11, AVR, ARM7/LPC2000 та інші широко розповсюджені мікропроцесори. Крім цього, програмне забазпечення містить понад 6000 цифрових і аналогових моделей різноманітних пристроїв. Proteus підтримує роботу з різними компіляторами та асемблерами. З допомогою

Proteus ISIS можна виконати симуляцію і відладку досить складних пристроїв, які можуть бути побудовані на декількох мікроконтролерах різних сімейств. Будь-яка симуляція електронних схем не відтворює абсолютно точно роботу реального пристрою. Щоб виконати відладку алгоритму роботи мікроконтролера, цього є достатньо. В Proteus є велика бібліотека моделей електронних компонентів і є змога самому створити відсутню модель. Якщо якийсь компонент нема можливості запрограмувати, то з сайту виробника можна завантажити SPICE – модель для нього і додати її у потрібний корпус.

Програмне забезпечення Proteus має додаткові компоненти COMPIM та USBCONN. Компонент COMPIM дозволяє віртуальному пристрою, створеного користувачем, підключитися до реального COM - порта комп’ютера. Компонент USBCONN дозволяє підключитися до реального USB – порта комп’ютера [2].

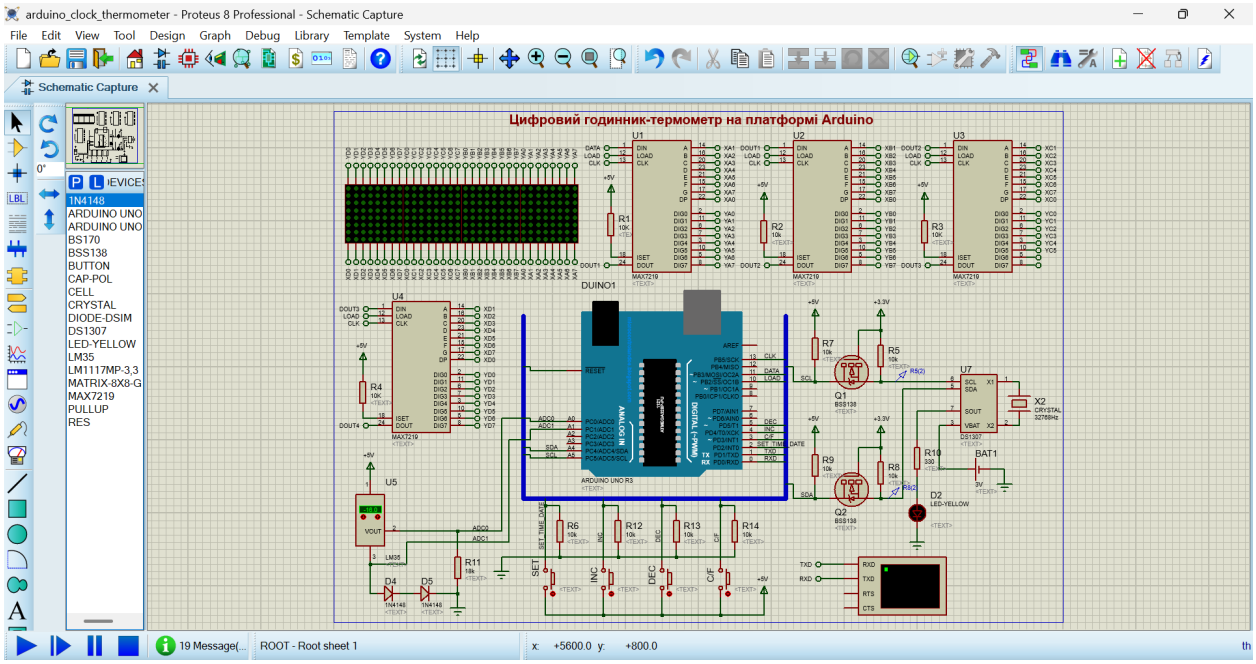


Рис.2.1. Середовище Proteus ISIS

Proteus є сумісним з популярними середовищами розробки вбудованого програмного забезпечення такими як [4]:

– CodeVisionAVR (тільки мікроконтролери AVR).
– IAR.
– ICC (МК AVR, msp430, ARM7).

- WinAVR (МК AVR).
- Keil (МК 8051 і ARM).
- HiTECH (МК 8051 і PIC).

2.2. Середовище розробки ПЗ під платформу Arduino – Arduino IDE

Arduino IDE – це програмне середовище розробки, яке використовує мову C++ і призначене для програмування всіх плат сімейства Arduino (рисунок 2.2). Arduino IDE дозволяє створювати програми з допомогою зручного текстового редактора, компілювати їх в машинний код і завантажувати на всі версії плати Arduino [5]. Arduino IDE дозволяє створювати та редагувати програмний код для плат на базі мікроконтролерів ATmega328, ATmega32U4, ATmega2560, AT91SAM3X8E, SAMD21, ESP8266, Intel x86, ATtiny85, STM [7].

Програма є повністю безкоштовною, завантажити її можна на офіційному сайті Arduino: https://www.arduino.cc/en/software/. Остання версія Arduino IDE на сьогодні 2.0.3. На сайті arduino.cc є декілька версій Arduino IDE для операційних систем Windows, Mac OS та Linux.

Програмне забезпечення Arduino IDE завоювало свою популярність серед багатьох користувачів завдяки своїй простоті, зручності та багатому набору готових бібліотек, а також підтримкою різних мікроконтролерів, список яких постійно зростає [5].

З допомогою Arduino IDE можна створювати скетчі як для простих блималок, таймерів та перемикачів навантажень, так і для систем розумного будинку, верстатів з ЧПК та квадрокоптерів. Інтерфейс Arduino IDE досить простий в освоєнні, основою є C++ подібна мова програмування з певними функціями. Для програмування Arduino використовується спрощена версія мови C++. Як і в інших C – подібних мовах програмування є ряд правил написання коду. Так само як і C++ мова є жостко типізованою і компільованою. Середовище Arduino IDE має меню, яке складається з вкладок File, Edit, Sketch, Tools і Help. Вкладка File містить пункти:

- New – для створення нового проекту.
- Open – для відкриття раніше збережених проектів.
- Open Recent – видає 10 останніх проектів з якими працювали.
- Sketchbook – як і вкладка Open для відкриття раніше збережених проектів.
- Examples – містить приклади готових проектів, які можна використовувати в такому вигляді як є, або попередньо відредагувати під свої потреби.
- Close – для закриття програмного проекту (скетчу).
- Save і Save as – для збереження скетчу.
- Page Setup – налаштування сторінки для друку на принтері.
- Print – для виводу друку на принтер.
- Preferences – налаштування програми Arduino IDE.
- Quit – для виходу з програми Arduino IDE.

У вкладці Preferences є пункт Sketchbook location, що вказує шлях для зберігання проектів. Пункт Editor Language служить для вибору мови інтерфейсу (доступно понад 60 мов), і різні пункти, які можна налаштувати на свій погляд, поставивши галочки в необхідних місцях. Також в пункті Additional Boards Manager URLs можна додати додаткові посилання для роботи зі сторонніми платами. Наприклад:

- для плат на базі ESP8266: http://arduino.esp8266.com/stable/-package_esp8266com_index.json;
- для плат на базі ESP32: https://dl.espressif.com/dl/package_esp32_index.json;
- для плат на базі STM32: http://dan.drown.org/stm32duino/-package_STM32duino_index.json.

Після зміни налаштувань потрібно натиснути OK, і якщо було змінено мову або масштаб інтерфейсу то потрібно перезавантажити Arduino IDE, щоб зміни вступили в силу.

Вкладка Edit:

- Undo – крок назад. Redo – крок вперед.
- Інші пункти копіювання, вибору, вставки, редагування, пошуку: Cut, Copy, Copy for Forum, Copy as HTML, Paste, Select All, Go to line…, Comment/Uncomment, Increase Indent, Decrease Indent, Increase Font Size, Decrease Font Size, Find…, Find Next, Find Previous.

Вкладка Sketch містить пункти:

- Verify/Compile – для перевірки скетчу на наявність помилок.
- Upload (Завантаження) – для завантаження скетчу в плату.
- Upload Using Programmer (Завантаження з допомогою програматора) – для завантаження скетчу з допомогою програма тора, при цьому у вкладці Tools потрібно вибрати Port (COM) і у вкладці Tools потрібно вказати програматор Programmer, який буде використовуватися.
- Export compiled Binary (Експорт бінарного файлу) – якщо відкрити потрібний проект і вибрати цей пункт, то проект буде відконвертовано з формату INO у формат HEX, який складається з двох файлів, і буде збережений в тій самій папці, в якій знаходиться основний проект.
- Show Sketch Folder – відриває папку зі скетчом.
- Include Library – якщо в проект необхідно додати бібліотеки, то вибрати їх можна тут, також можна додати ZIP – архів, який містить бібліотеки або завантажити з Інтернету використовуючи вкладку Manage Libraries…
- Add File… – додає файл до скетчу (він буде скопійований з поточного місця). Новий файл появиться в новій вкладці (закладці) вікна скетчів. Файл може бути видалений з числа скетчів через меню закладок (вкладок).

Вкладка Tools містить пункти:

- Auto Format – натиснувши на цей пункт Arduino IDE автоматично проставить відступи і пропуски, після чого, скетч стає краще читабельним.
- Archive Sketch – служить для архівування скетчу в ZIP – архів. Архів зберігається в тій самій папці, що і скетч.
- Fix Encoding & Reload – виправляє розходження між картою символів редактора Arduino IDE і картами символів інших ОС.

- Serial Monitor – відкриває монітор порту для відправки і отримання даних від плати, яка підключена до ПК. Швидкість передачі даних повинна бути така сама, яка вказана у скетчі.
- Serial Plotter – цей інструмент використовується для виводу даних на графіку в режимі реального часу.
- WiFi101/WiFiNINA Firmware Updater – ця функція для роботи з модулями Wi-Fi і потрібна для прошивки в плату SSL сертифікату.
- Board – вибирається плата, в яку буде завантажено скетч. Якщо в списку відсутня плата, то її можна додати через Boards Manager…(Менеджер плат). Якщо використовується плата сторнніх виробників, то можливо буде потрібно додати додаткові посилання (зсилки) у вкладці File пункт Preferences.
- Port – служить для вибору послідовного COM – порта, до якого підключена плата.
- Get Board Info – служить для отримання інформації про підключену плату.
- Programmer – при використанні програма тора для завантаження скетчу, в цьому пункті слід вибрати модель програма тора. При цьому потрібно вибрати порт і щоб завантажити скетч вибрати вкладку Sketch -> Upload Using Programmer.
- Burn Bootloader – завантаження загружчика (bootloader). Bootloader – це мікропрограма, яка потрібна для взаємодії плат Arduino з програмою Arduino IDE. Ця функція може знадобитися, наприклад, якщо придбано порожній мікроконтролер і в подальшому він має використовуватися для прошивки скетчів Arduino IDE, або для перепрошивки загружчика. Для запису загружчика (bootloader) використовується програматор.

  Arduino IDE має також панель інструментів, яка містить 6 кнопок:

 Verify – виконується перевірка скетча шляхом компіляції.

 Upload – виконує завантаження скетчу в плату, і при утриманні кнопки Shift активується друга функція цієї кнопки – Upload Using Programmer (Завантажити з допомогою програматора).

 New – створює новий скетч Arduino IDE.

 Open – відкриває список раніше збережених проектів.

 Save – зберігає проект, і при утриманні кнопки Shift активується друга функція цієї кнопки – Save as…

Serial Monitor – відкриває вікно монітор порту для обміну даними між комп’ютером і підключеною до нього платою.

В низу під панелею інструментів знаходиться панель вкладок, яка призначена для роботи з проектами, що складаються з декількох файлів. Складається панель з вкладок і випадаючого меню для управління вкладками. Arduino IDE підтримує розширення *.ino,*.c, *.cpp, *.h.

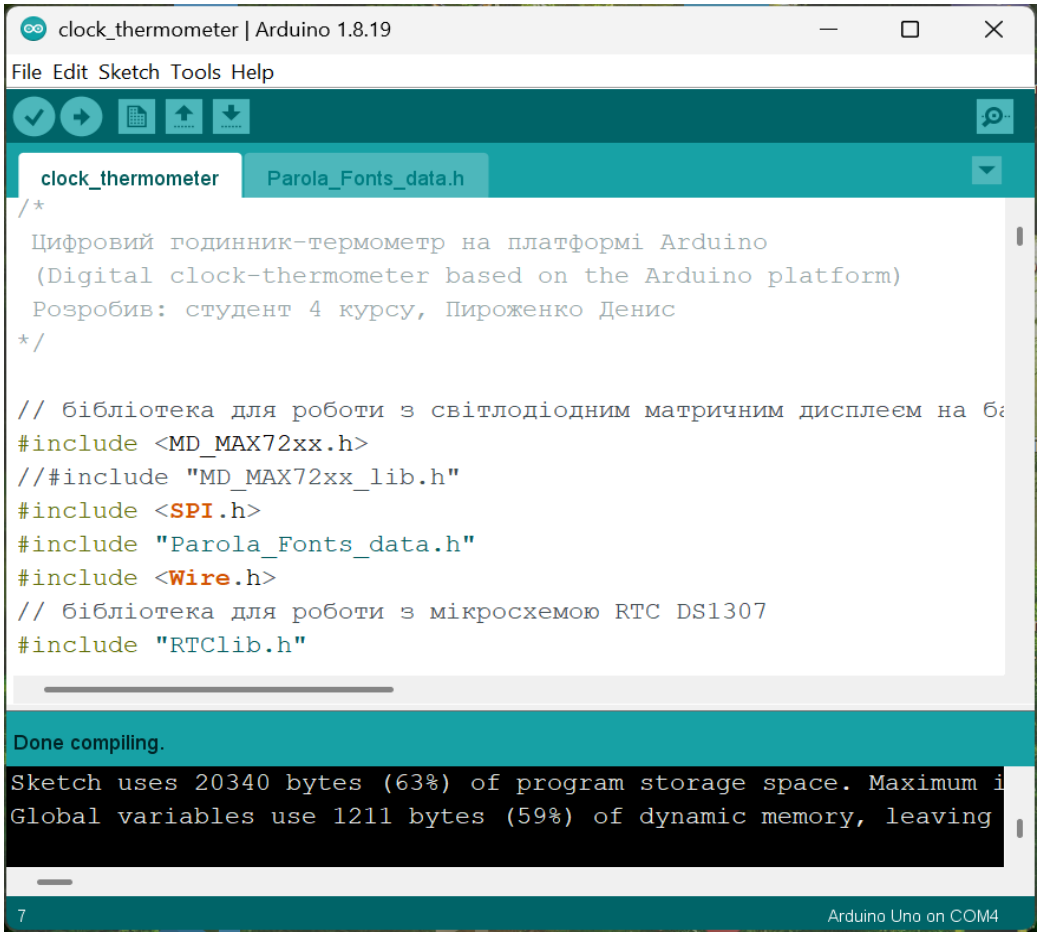


Рис.2.2. Середовище розробки Arduino IDE

РОЗДІЛ ІІІ
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ГОДИННИКА-ТЕРМОМЕТРА

3.1 Вибір елементної бази для проектування цифрового годинника-термометра

Для проектування цифрового годинника-термометра було вибрано наступні електронні комплектуючі:

- Плата Arduino Micro на мікроконтролері AVR ATmega32u4;
- Прецизійний давач температури за Цельсієм LM35DZ;
- Мікросхема годинника реального часу RTC DS1307;
- Дисплейнй модуль FC-16 з 4 точкових світлодіодних матриць на базі мікросхем – драйверів MAX7219;
- 4 двохконтактні кнопки на 4 виводи 6×6×5 мм.

Огляд платформи Arduino і плати Arduino Micro на МК ATmega32u4

В Інтернет - магазинах сучасних електронних комплектуючих можна знайти велике різноманіття мікроконтролерних платформ, які мають різні архітектури, надійність, габарити та ціну. Як одна з самих популярних і легких для вивчення є мікроконтролерна платформа Arduino.

Arduino представляє собою апаратно-програмний комплекс, який призначений для розробки електронних програмовних пристроїв. Апаратна складова представлена різними платами, які називаються Arduino Uno R3, Arduino Micro, Arduino Mega2560 та інші. Завдяки повністю відкритій архітектурі платформи Arduino, випуском і доповненням лінійки її плат займаються окрім офіційних виробників також і сторонні. Програмною складовою Arduino є безплатне програмне забезпечення Arduino IDE. Arduino IDE використовується для написання програмного коду, його компіляції і

прошивки у МК плати Arduino. Детальніший опис програмного забезпечення Arduino IDE наведено в розділі “Засоби розробки апаратного та програмного забезпечення цифрового годинника-термометра” [5].

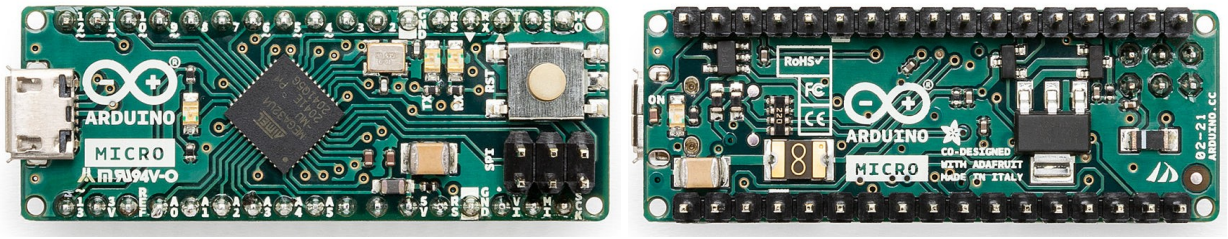


Рис.3.1. Плата Arduino Micro на МК ATmega32u4

Плата Arduino Micro представляє собою пристрій на мікроконтролері ATmega32u4, який розроблено за участі компанії Adafruit Industries. Arduino Micro – це мініатюрна плата, яка суміщає в собі всі необхідні можливості, простоту у використанні і низьку вартість (рисунок 3.1). На платі знаходяться 20 цифрових виводів входу/виходу, 7 цифрових виводів можуть бути задіяні для виводу ШІМ – сигналу (PWM – сигналу), 12 аналогових виводів (входи АЦП), кварцовий генератор на 16 МГц, гніздо micro-USB, роз’єм ICSP і кнопка reset. Плата має все необхідне для роботи з МК. Завдяки її форм - фактору плата легко встановлюється на макетній платі [2].

Micro є схожою до плати Arduino Leonardo в тому, що МК ATmega32u4 має вбудовану підтримку USB - підключення, завдяки чому непотрібно допоміжного процесора, і може розпізнаватися на комп’ютері в якості периферійного пристрою (клавіатура, джойстик, геймпад або комп’ютерна мишка) у доповнені до віртуального (CDC) послідовного порта (COM).

Технічні характеристики плати Arduino Micro

- Мікроконтролер: AVR ATmega32u4.
- Тактова частота: 16 МГц.
- Вхідна напруга живлення: 7 - 12 В.
- Напруга логічних рівнів: 5 В.
- Цифрових портів загального призначення: 20.
- Максимальний струм з порта виводу: 40 мА.

- Максимальний вихідний струм 5 В: 800 мА.
- Максимальний вихідний струм 3,3 В: 50 мА.
- Портів з ШІМ: 7.
- Портів з АЦП: 12.
- Розрядність АЦП: 10 біт.
- Об’єм Flash – пам’яті: 32 КБ.
- Об’єм EEPROM – пам’яті: 1 КБ.
- Об’єм оперативної пам’яті SRAM: 2,5 КБ.

Підключення живлення до плати Arduino Micro

Плата Arduino Micro може живитися через порт micro-USB від комп’ютера, павербанка або від блока живлення підключеного до розетки. Джерело живлення буде вибрано автоматично. Також вивід +5V являється не тільки виходом, але і входом. На нього можна подавати напругу і пристрій буде працювати при умові, що напруга строго рівна 5 В. Також можна подавати напругу від 6 до 20 В на вивід VIN. При подачі напруги 20 В на платі буде сильно грітися стабілізатор, що може привести до виходу з ладу плату. Якщо подавати 5 В, то плата взагалі може не запрацювати. Якщо і запрацює, то на цифрових виводах напруга буже нижчою 5 В. Це зв’язано з тим, що стабілізатор напруги не має 100 % ККД. Рекомендована напруга для живлення чере вивід VIN – від 7 до 12 В [4].

Arduino Micro: живлення плати від зовнішнього джерела:

- **5V** – на вивід подається напруга 5 Вольт.
- **3.3V** – на вивід подається напруга 3,3 Вольта.
- **GND** – спільне заземлення (вивід землі).
- **Vin** – вивід служить для подачі напруги.
- **IREF** – вивід служить для надання інформації про напругу плати.

Плата Arduino Micro підтримує три типи пам’яті:

- Flash – пам’ять об’ємом 32 КБайти, яка використовується для зберігання скетчів. Коли плата прошивається, скетч записується саме у Flash – пам’ять.

- SRAM – пам’ять – оперативна пам’ять об’ємом 2,5 КБайти. В цій пам’яті зберігаються змінні, які створюються в скетчі, при відключенні живлення всі дані пропадають.
- EEPROM – енергонезалежна пам’ять об’ємом 1 КБайт. В цій пам’яті можуть зберігатися різні дані, які не пропадуть при відключенні живлення плати.

Входи і виходи на платі Arduino Micro

Плата Arduino Micro, як було зазначено вище, має 20 цифрових виводів. Вони можуть бути налаштовані (запрограмовані) як вхід або вихід. Робоча напруга цих виводів становить 5 В. Кожний з них має підтягуючий резистор і подана на один з цих виводів напруга нижче 5 В все рівно (все одно) буде вважатися як 5 В (логічна одиниця). Аналогові входи A0 – A5, A6 – A11 (на цифрових виводах 4, 6, 8, 9, 10 і 12). Всього Micro має 12 аналогових входів, причому входи з A0 по A5 марковані безпосередньо на виводах, а інші, до яких можна отримати доступ в програмі з використанням констант з A6 по A11, розподілені відповідно на цифрових виводах 4, 6, 8, 9, 10 і 12. Всі вони також можуть використовуватися в якості цифрових входів/виходів. Через них вимірюється напруга, що подається, і повертається значення від 0 до 1024 з допомогою програмної функції analogRead(). Виміряти напругу можна з точністю 0,005 В.

Широтно – імпульсна (ШІМ) модуляція Arduino Micro. Плата Arduino Micro має 7 виводів ШІМ, це виводи 3, 5, 6, 9, 10, 11 і 13. Для використання ШІМ в Arduino використовується спеціальна функція analogWrite().

Виводи (піни) 0 (RX) і 1 (TX) використовуються для обміну даними по послідовному інтерфейсу. Виводи для комунікації по інтерфейсу SPI не підключені до цицрофвих виводів. Також на виводі D13 є вбудований в плату світлодіод. Мікроконтролер ATmega32U4 також підтримує інтерфейси I^2C (TWI) і SPI. Програмне забезпечення Arduino IDE включає бібліотеку Wire, яка спрощує використання шини I^2C. Для інтерфейсу SPI використовується бібліотека SPI.

Плата Arduino Micro має 4 виводи для обробки зовнішніх переривань: 3 (INT 0), 2 (INT 2), 0 (INT 3) і 1 (INT 4). Виводи переривань можуть бути сконфігуровані для виклику переривання і функції його обробки. Для задання функції, яку необхідно викликати при виникненні зовнішнього переривання, використовується функція attachInterrupt(interrupt, function, mode). У функції attachInterrupt параметр interrupt є номером переривання, function – функція, що викликається при виникненні переривання. Функція function немає параметрів і не повертає ніяких значень. Така функція називається обробником переривання. Параметр mode визначає умову, при якій повинна спрацьовувати переривання. Він може приймати одне з визначених значень: LOW – переривання буде спрацьовувати кожний раз, коли на виводі присутній низький рівень сигналу, CHANGE – переривання буде спрацьовувати кожний раз, коли міняється стан виводу, RISING – переривання спрацює, коли стан виводу зміниться з низького рівня на високий, FALLING – переривання спрацює, коли стан виводу зміниться з високого рівня на низький; FALLING – переривання спрацює, коли стан виводу зміниться з високого рівня на низький, HIGH – переривання буде спрацьовувати кожний раз, коли на виводі присутній високий рівень сигналу (тільки для Arduino Due).

- **Інтерфейс SPI: виводи на роз'ємі ICSP.** З використанням бібліотеки SPI ці виводи дозволяють здійснювати зв'язок по інтерфейсу SPI. В Arduino Micro лінії SPI виведені тільки на роз'єм ICSP і на окремі виводи MISO, MOSI, і SCK, які розміщені поряд з ним. Вони не підключені до ніяких цифрових виводів (входів/виходів).
- **RX_LED/SS.** Цей вивід з'єднаний з світлодіодом RX_LED, який візуально показує активність процесу передачі даних через USB. Проте разом з тим, він може використовуватися в якості виводу SS при роботі з SPI - інтерфейсом.

Окрім перерахованих виводів на платі є ще декілька виводів:

- **Вивід AREF.** Опорна напруга для аналогових входів, яка може бути задіяна функцією analogReference().

- **Вивід Reset.** Формування низького рівня (LOW) на цьому виводі приведе до перезавантаження мікроконтролера. Зазвичай цей вивід служить для функціонування кнопки скидання (Reset) на платах розширення.

На рис.3.2 зображено розміщення та функціональне призначення виводів на платі Arduino Micro.

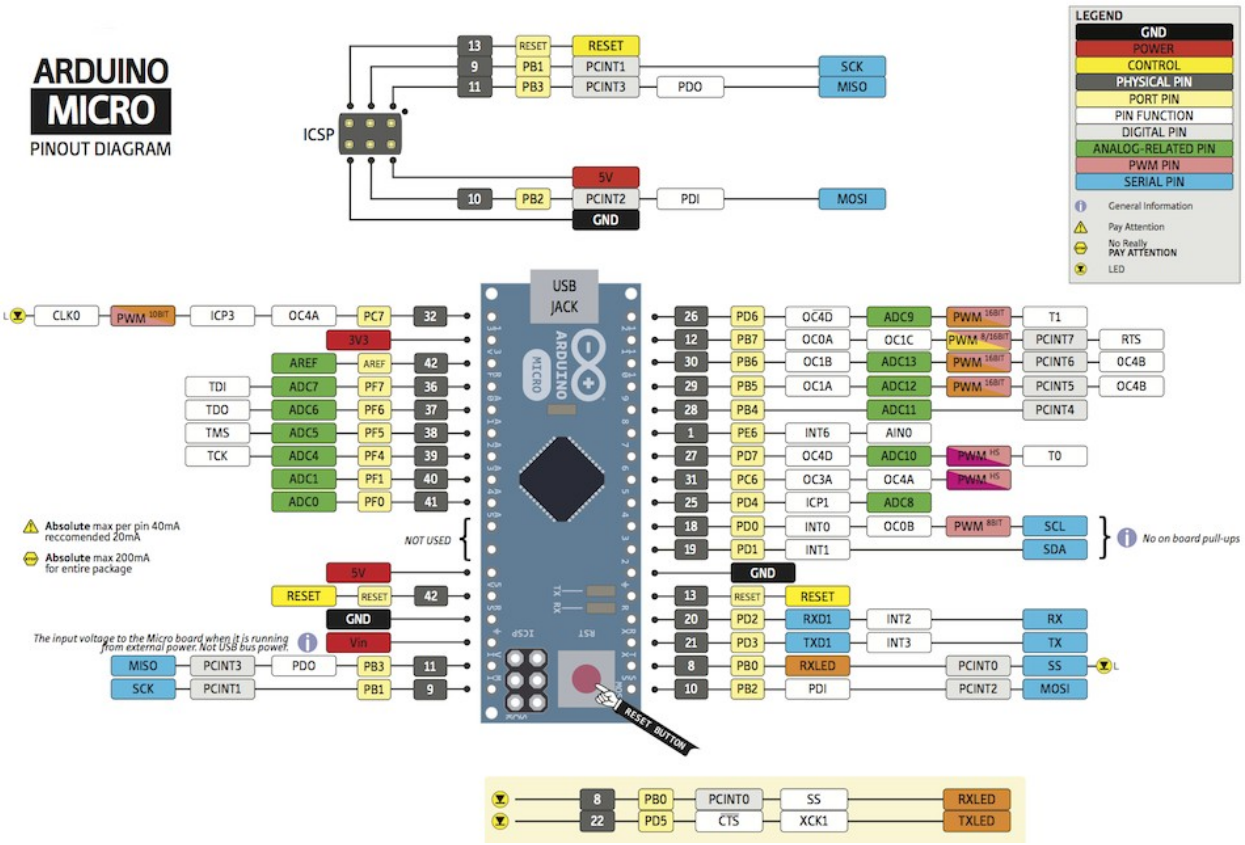


Рис.3.2. Розміщення та функціональне призначення виводів на платі Arduino Micro

Програмування МК плати Arduino Micro

Плату Arduino Micro програмують з допомогою ПЗ Arduino IDE. Для цього в Arduino IDE через Tools -> Board потрібно вибрати плату Arduino Micro [3].

В мікроконтролер ATmega32U4 плати Arduino Micro вже прошито завантажувач (bootloader), який дозволяє завантажити прошивку без допомоги зовнішнього апаратного програматора. Для зв’язку використовується протокол AVR109. Також можна обійти bootloader і запрограмувати МК через роз’єм ICSP (In-Circuit Serial Programming).

Автоматичне (програмне) скидання та ініціалізація bootloader'а

Щоб кожний раз перед завантаженням не треба було натискати кнопку скидання, Arduino Micro спроектована таким чином, що дозволяє виконувати її скидання програмно з комп'ютера. Скидання спрацьовує після закриття віртуального COM – порта, який попередньо був відкритий на швидкості 1200 бод. При спрацюванні цієї умови, мікроконтролер скинеться, розірвавши USB - підключення з комп'ютером (при цьому віртуальний COM - порт зникне). Після перезавантаження мікроконтролера, запускається bootloader (завантажувач), який залишається активним приблизно протягом 8 секунд. Окрім цього, запустити завантажувач (bootloader) можна натиснувши кнопку скидання на платі Micro. Варто відмітити, що при першому включенні пристрою замість запуску завантажувача, мікроконтролер відразу перейде до виконання програми користувача (при її наявності).

Через особливості механізму скидання плати Arduino Micro, рекомендується надавати програмному забезпеченню можливість виконувати її скидання перед завантаженням програми. Якщо програмне забезпечення не взмозі скинути пристрій, то завжди можна запустити bootloader вручну натиснувши кнопки скидання на платі.

Захист USB - порта від струмових перевантежень

В Arduino Micro є відновлювальний запобіжник, що захищає USB - порти комп'ютера від короткого замикання і струмових перевантажень. Не дивлячись на те, що більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий захист. Якщо від USB - порта буде споживатися струм більше 500 мА, запобіжник автоматично розірве з'єднання до усунення причин короткого замикання або перевантаження.

Фізичні характеристики плати Arduino Micro

Arduino Micro має наступні розміри: довжина 48 мм і ширина 18 мм. Роз'єм USB трохи виходить за межі друкованої плати. Arduino Micro важить всього 12 грам. Плата має 4 отвори для можливості її кріплення на поверхні. Відстань між виводами рівна 2,54 мм.

Модуль годинника реального часу (Real Time Clock) DS1307

Модуль RTC (Real Time Clock) годинника реального часу DZ1307 є високоточним енергонезалежним модулем, який можна використовувати в таких проектах як годинник, будильник, секундомір і т.д., а також з його допомогою можна запустити потрібні процеси за розкладом. Модуль побудований на мікросхемі DS1307Z, яка має I^2C інтерфейс (),

Модуль DS1307 може працювати протягом 5 років від літієвої батарейки CR2032-210mAh. Крім мікросхеми годинника реального часу модуль містить мікросхему I^2C EEPROM 24C32 та інтерфейс для підключення давача температури DS18B20. Чіп енергонезалежної пам'яті EEPROM дозволяє зберігати дані отримані від давача локально на модулі без необхідності постійно використовувати мікроконтролер. Мікросхема DS1307 має програмований генератор прямокутних імпульсів, що дозволяє генерувати одну з чотирьох частот (1 Гц, 4096 Гц, 8192 Гц або 32768 Гц), яка в свою чергу може використовуватися для корекції похибки кварцового резонатора. Модуль годинника реального часу DS1307 має два ряди контактів – P1 і P2, а також слот для встановлення батарейки живлення CR2032. Ряд P1 має контакти SQ, DS, SCL, SDA, VCC, GND, BAT. Ряд P2 має контакти DS, SCL, SDA, VCC, GND. Живлення модуля може відбуватися від контролера Arduino, іншого управляючого мікропроцесорного пристрою або від зовнішнього джерела живлення. Мікросхема DS1307 має вбудований блок, який визначає аварійне відключення живлення і автоматично підключає резервну батарею. При цьому відлік часу продовжується і після відновлення живлення годинник показує правильний час.

Технічні характеристики модуля годинника реального часу DS1307:

- підключення модуля: інтерфейс I^2C;
- тактова частота шини I^2C: до 100 кГц;
- рівень "0" на шині I^2C: -0,5…0,8 В;
- рівень "1" на шині I^2C: 2,2…VCC В;
- відлік часу: година, хвилина, секунда AM/PM;

- відлік дати: рік, місяць, день (враховує високосні роки);
- точний календар до 2100 року;
- доступна пам’ять: 56 байт енергонезалежної пам’яті;
- батарейка живлення: CR2032 (напруга живлення батареї 2,0…3,5 В (номінальна 3 В));
- живлення модуля: 5 В;
- струм споживання (в режимі очікування): до 200 мкА;
- струм споживання (під час передачі даних): до 1,5 мА;
- струм споживання (під час резервного живлення): 300…800 нА;
- робоча температура: 0…70 °С;
- габарити модуля: 28x25x8.

DS1307 є невеликим модулем, що виконує відлік часу (Рис.3.11). Він побудований на мікросхемі годинника реального часу DS1307ZN і має слот для літієвої батарейки (LIR2032/CR2032-210 mAh), яка дозволяє автономно живити мікросхему годинник реального часу протягом тривалого часу. Модуль також має енергонезалежну пам’ять EEPROM на 32 КБ (AT24C32). Мікросхеми пам’яті AT24C32N і годинника реального часу DS1307Z пов’язані спільною шиною I^2C [4].

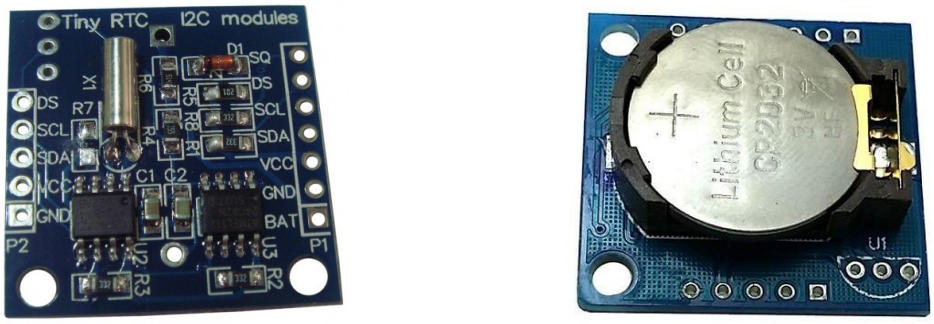


Рис.3.3. Модуль годинника реального часу DS1307

Технічні характеристики модуля DS1307:

- напруга живлення: 5 В;
- робоча температура: -40 °С…+85 °С;
- пам’ять: 56 байт (енергозалежна);

- батарейка/акумулятор: LIR2032 (автоматичне визначення джерела живлення);
- інтерфейс: I^2C;
- габарити; 28 мм×25 мм×8 мм.

Використання модуля DS1307 часто є виправданим, наприклад, коли дані зчитуються рідко, інтервалом більше тижня, використовувати власні ресурси мікрконтролера, невиправдано чи неможливо. Забезпечення безперебійного живлення, наприклад плати Arduino, на тривалий термін дорого, навіть при використанні батарейки.

Модуль підключається до апаратної шини I^2C використовуючи виводи SDA, SCL, VCC і GND. Вивід 32K є виходом меандру з фіксованою частотою 32768 кГц. Вивід SQW може використовуватися як вивід переривання, або як вихід меандра з програмованою частотою. При наявності основного (2,3 < VCC < 5,5 В) і резервного (2,3 < Vbat <5,5 В) джерела живлення, модуль працює від основного джерела живлення. При відсутності резервного живлення (Vbat), модуль працює від основного джерела живлення. При відсутності основного живлення (Vcc < Vpf і Vcc < Vbat), модуль працює від резервного (продовжує відраховувати час, але не відповідає на запити по шині I^2C). При відсутності основного (Vcc) і резервного (Vbat) живлення, модуль відключається і всі його регістри скидаються. Модуль побудований на мікросхемі DS1307Z, який має: інтерфейс I^2C (частота 100 кГц, адреса 0x68), 64 однобайтними регістрами (56 з яких доступні для зберігання даних користувача), компаратором (для переключення між основним і резервним живленням), блоком підзарядки акумуляторної батареї. Модуль дозволяє зчитувати секунди, хвилини, години, дні тижня, місяць і рік.

Комунікація з іншими пристроями здійснюється по шині I^2C через виводи SCL і SDA. Для зменшення шумів по лінії живлення використовуються конденсатори C1 та C2. Резистори R2 і R3 підтягнуті до лінії живлення і забезпечують належні рівні сигналів SCL і SDA.

Щоб перевірити дієздатність модуля на 7-ий вивід мікросхеми DS1307Z подається SQ - сигнал прямокутної форми з частотою 1 Гц. Резистори R4, R5, R6 і діод VD1 необхідні для зарядки літієвого акумулятора.

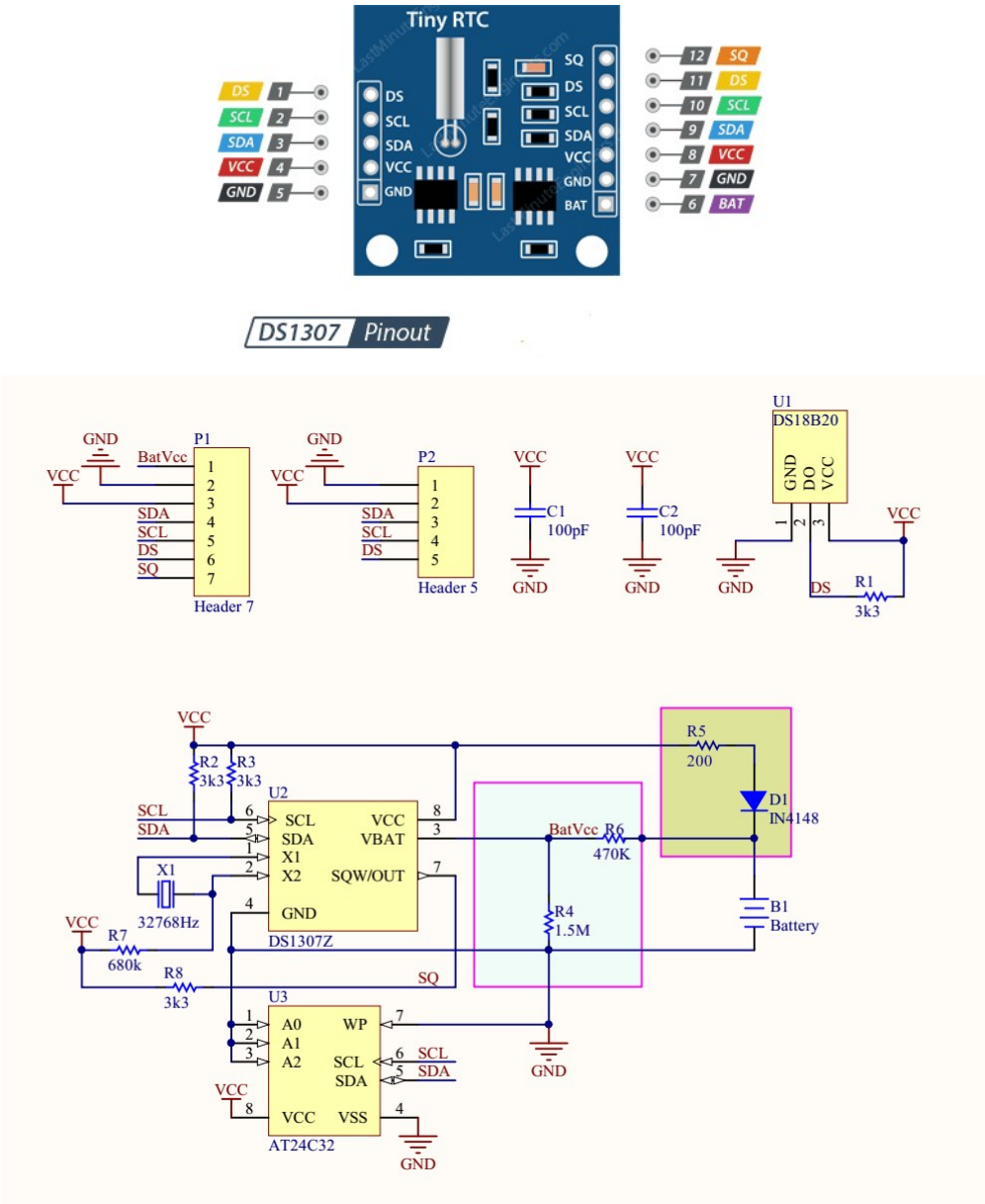


Рис.3.4. Принципова схема і призначення виводів модуля DS1307

Також, на платі є посадкове місце U1 для встановлення цифрового давача температури DS18B20. Покази з давача температури можна зчитувати через вивід DS, який підтягнуто до лінії живлення через резистор R1 3,3 кОм. Схема принципова та призначення виводів модуля DS1307 показано на рисунку 3.4.

На платі розташовано дві групи контактів, кроком 1 дюйм (2,54 мм), для зручного підключення до макетної плати, в які впаюються штирьові роз'єми.

Перша група контактів:

- DS: вивід DS18B20 (1-wire);
- SCL: лінія тактування (Serial Clock);
- SDA: лінія даних (Serial Data);
- VCC: “+” живлення модуля;
- GND: “-” живлення модуля.

Друга група контактів:

- SQ: вхід 1 МГц
- DS: вивід DS18B20 (1-wire);
- SCL: лінія тактування (Serial Clock);
- SDA: лінія даних (Serial Data);
- VCC: “+” живлення модуля;
- GND: ”-” живлення модуля;

Підзарядка акумулятора LIR2032

В модулі передбачена зарядка акумуляторної батарейки (LIR2032) з допомогою резисторів R4, R5, R6 і діода D1. В цій схемі є недолік, через резистор R4 і R6 відбувається невеликий розряд акумулятора. Для того щоб можна було використовувати звичайну батарейку CR2032 потрібно видалити схему живлення, для цього випаюємо резистори R4, R5, R6 і діод VD1, замість резистора R6 запаюємо перемичку (рисунок 3.5).

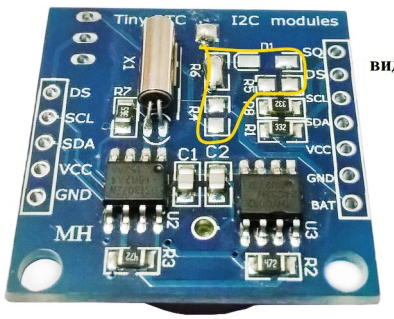


Рис.3.5. Модуль DS1307

Огляд прецизійного давача температури за Цельсієм LM35DZ

Серія давачів LM35 представляє собою прецизійні інтегральні давачі температури. Вихідна напруга давача є лінійно пропорційною температурі за Цельсієм [10]. Перевага пристрою (давача) LM35 над лінійними давачами

температури, відкаліброваними в Кельвінах, полягає в тому, що користувачу не потрібно віднімати велику постійну напругу з вихідного сигналу для отримання зручної шкали за Цельсієм. Пристрій LM35 не потребує будь-якого зовнішнього калібрування або налаштування для забезпечення типової точності ±¼ (1/4) °C при кімнатній температурі і ±¾°C на всьому діапазоні температур −55 °C до 150 °C. Більш низька вартість забезпечується калібруванням на рівні кристалу (підкладки пластини). Низький вихідний імпеданс, лінійний вихід і точне вбудоване калібрування пристрою LM35 робить взаємодію зі схемами зчитування або управління досить простою. Пристрій використовується з одиночним джерелом живлення, або з плюсовими і мінусовими джерелами живлення. Оскільки пристрій LM35 споживає лише 60 мкА від джерела живлення, він має дуже низький самонагрів – менше 0,1 °C в нерухомому повітрі. Пристрій LM35 розрахований на роботу в діапазоні температур -55 °C до 150 °C, а пристрій LM35C розрахований на діапазон від -40 °C до 110 °C (-10 °C з підвищеною точністю). Пристрої (давачі) серії LM35 доступні в герметичних корпусах транзисторів TO, а пристрої LM35C, LM35CA і LM35D доступні в пластикових корпусах транзисторів TO-92. Пристрій LM35D доступний в 8-вивідному малогабаритному корпусі для поверхневого монтажу і в пластиковому корпусі TO-220.

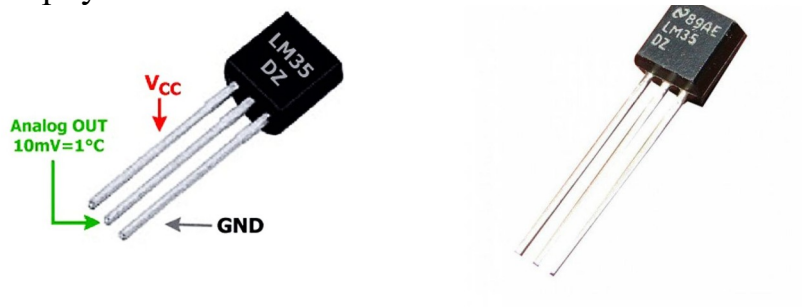


Рис.3.6. Давач температури LM35

Характеристики давача температури LM35:

- відкалібрований безпосередньо в градусах Цельсія;
- лінійний коефіцієнт перетворення + 10 мВ/°C;

- точність вимірювання 0,5 °C;
- діапазон вимірювання -55 °C до 150 °C;
- підходить для віддалених застосувань;
- низька вартість завдяки технології масового виробництва;
- діапазон робочої напруги від 4 В до 30 В;
- струм споживання не перевищує 60 мкА;
- низький самонагрів 0,08 °C в нерухомому повітрі;
- нелінійність: лише ±1/4 °C типове значення;
- вихід з низьким імпедансом 0,1 Ом для нагрузки 1 мА.

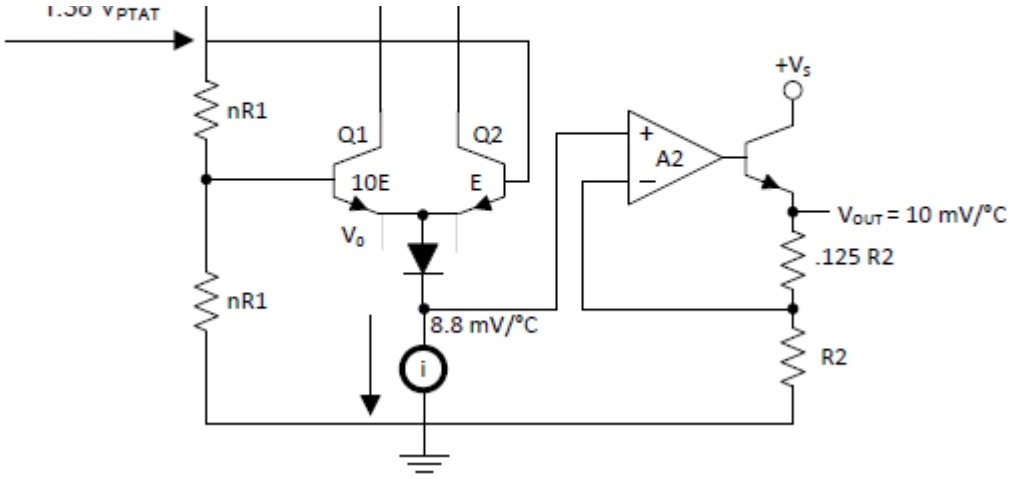


Рис.3.7. Функціональна блок-схема давача температури LM35DZ

Точність давача LM35 визначається по відношенню до простої лінійної передаточної функції: V_{OUT}= 10 мВ/°C × T, де V_{OUT} – вихідна напруга LM35, T – температура в °C.

Особливості LM35 дозволяють використовувати його в багатьох задачах (проектах) по вимірюванню температури, а різні корпуси розширюють його гнучкість (можливості).

Мікросхема давача LM35 має три виводи, два з яких призначені для живлення давача, а третій є виходом. Для отримання точних результатів давач LM35 не потребує будь-якого калібрування.

Переваги давача LM35: лінійна залежність вихідного сигналу (температура/напруга), низький вихідний опір, вбудована схема калібрування. Давач може працювати в діапазоні від -55 º до 150 º С.

Аналоговий сигнал на виході давача прямо пропорційний змінні температури в градусах Цельсія, і на кожний градус приходиться 10 мВ. Струм споживання давача становить близько 60 мкА, через це само розігрів давача LM35 становить всього 0,1 °С.

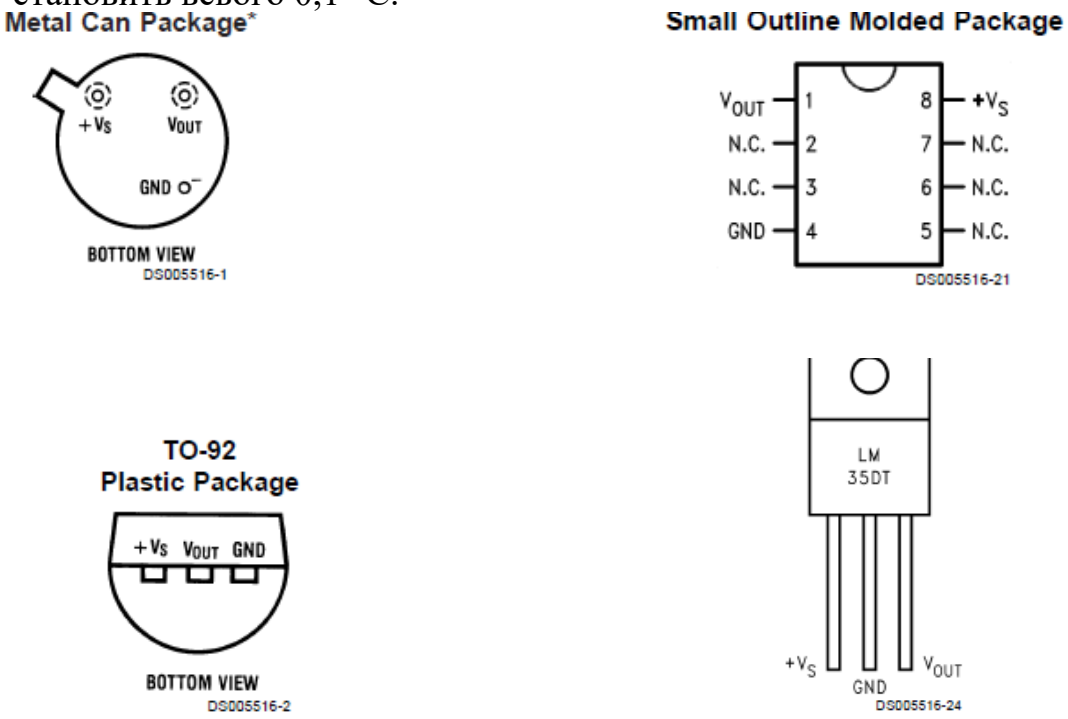


Рис.3.8. Розміщення і призначення виводів в різних корпусах давача LM35

В основному давач LM35 випускається в корпусі TO-92, але він також може бути в корпусі TO-220 або TO-46. Їх характеристики однакові, відмінність тільки в областях застосування. Наприклад, на відмінну від корпуса TO-92 давач в металічному корпусі TO-46 може бути використаний для контактного вимірювання температури поверхні. Давач в TO-92 використовується в основному для вимірювання температури повітря.

Областями застосування давача температури є блоки живлення, системи управління батареями (акумуляторами), системи опалення, вентиляції та кондиціювання повітря, побутова техніка (прилади).

Давач температури LM35 представляє собою інтегральну схему, яка призначена для вимірювання температури, використовується в пристроях, так або інакше зв'язаних з контролем температури. Давач LM35 є недорогою, надійною і досить точною мікросхемою. Похибка вимірювання становить ±0,5 °C.

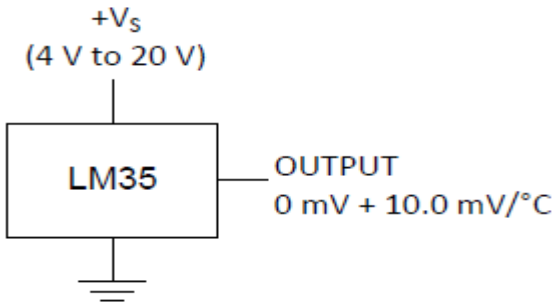


Рис.3.9. Типова схема включення давача за Цельсієм для вимірювання температури в діапазоні від 2 °C до 150 °C

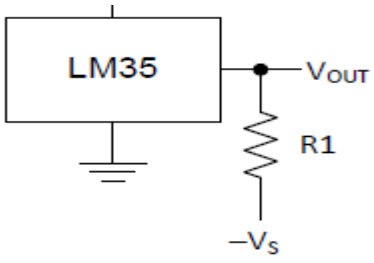


Рис.3.10. Схема включення давача температури на повний діапазон вимірювання (VOUT = +1,500 мВ при 150 °C = +250 мВ при +25 °C = -550 мВ при -55 °C

Спроможність давача управляти ємнісними навантаженнями

Як і більшість мікросилових схем, давач LM35 має обмежену спроможність управляти великим ємнісними нагрузками. Сам один пристрій LM35 здатний управляти 50 пФ без особливих мір обережності (рисунок 3.11). Якщо очікуються більш важкі навантаження, то легко ізолювати або розв'язати навантаження з допомогою резистора. Допуск ємності можна покращити з допомогою демпфера серії R-C з виходу до землі (рисунок 3.12).

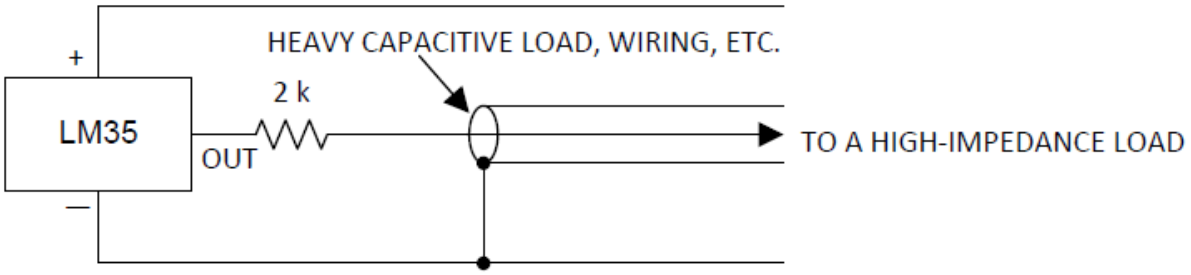


Рис.3.11. Схема розв’язки LM35 від ємнісної нагрузки

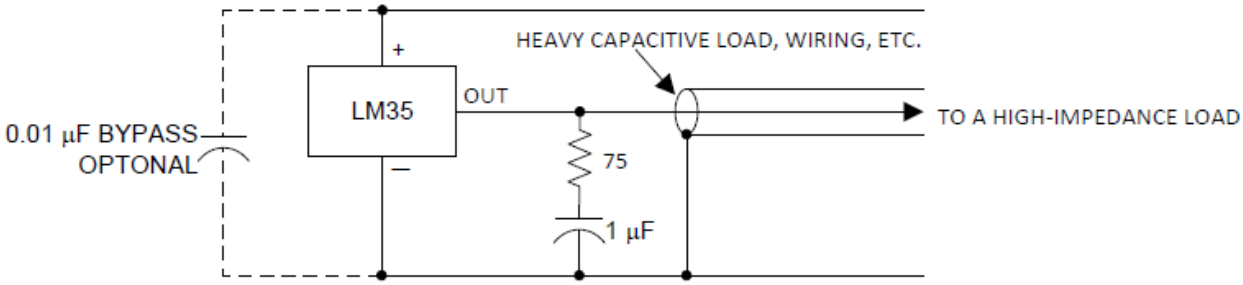


Рис.3.12. Схема розв’язки LM35 з R-C демпфером

Оскільки давач LM35 представляє собою простий давач температури з аналоговим виходом, вимоги до конструкції (проекту), які зв’язані з компоновкою є важливішими ніж електричні вимоги.

На рисунку 3.13 зображено графіки залежності точності давача від температури.

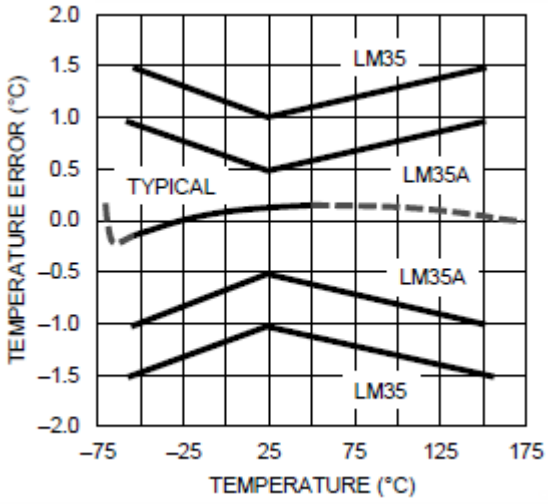


Рис.3.13. Залежність точності від температури

На рисунках нижче 3.14 – 3.23 показано типові схеми включення давача температури LM35.

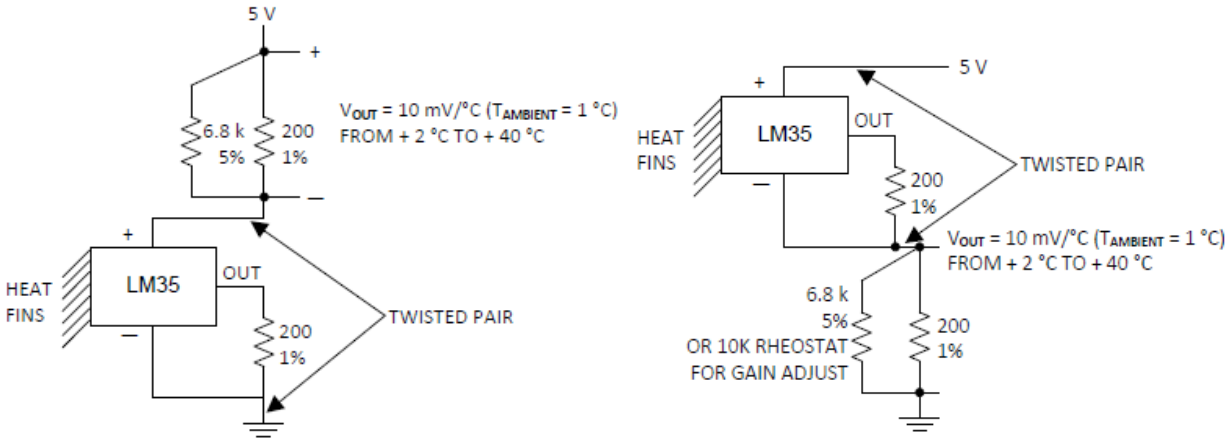


Рис.3.14. Схеми включення двохпровідного дистанційного давача температури

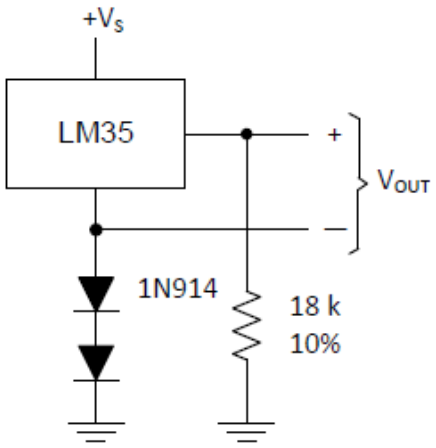


Рис.3.15. Схема включення давача температури з одним живленням для вимірювання в діапазоні -55 °C до +150 °C

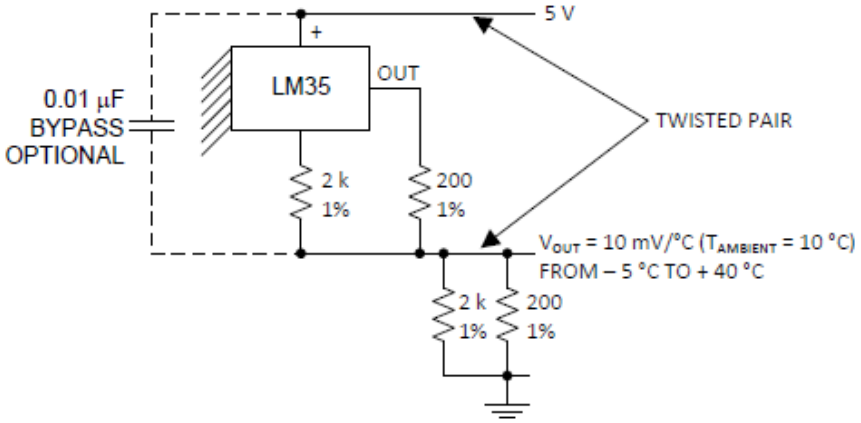


Рис.3.16. Схеми включення двохпровідного дистанційного давача температури (заземлений давач)

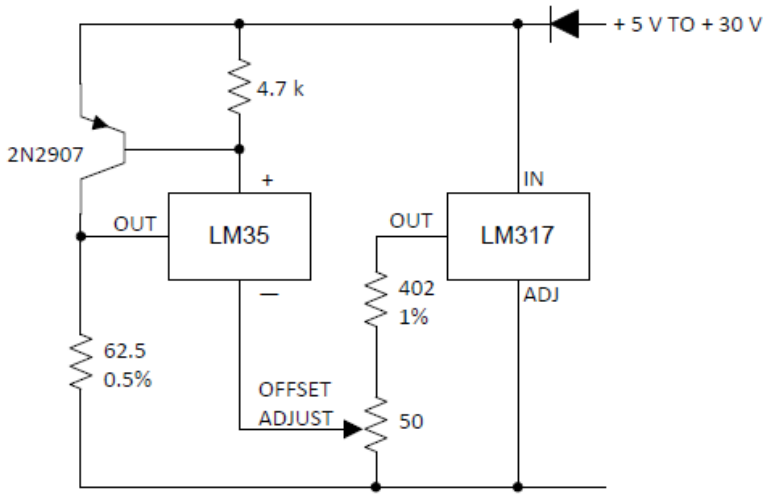


Рис.3.17. Джерело струму 4-до-20 мА (від 0 °C до 100 °C)

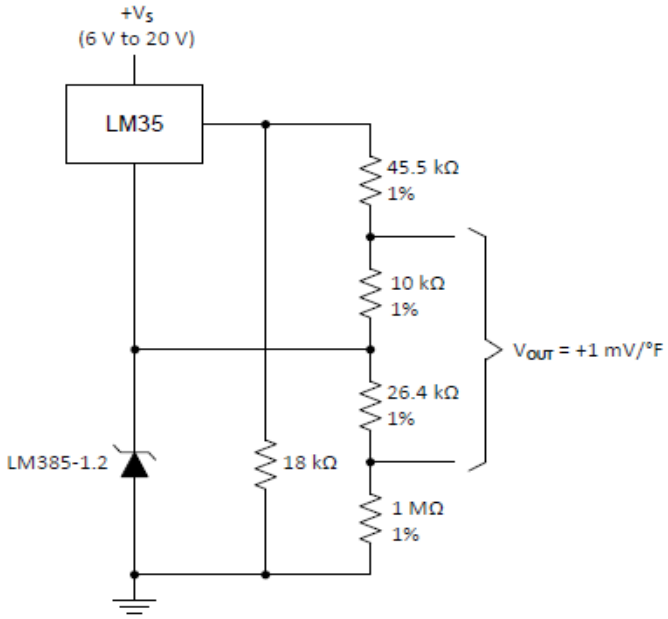
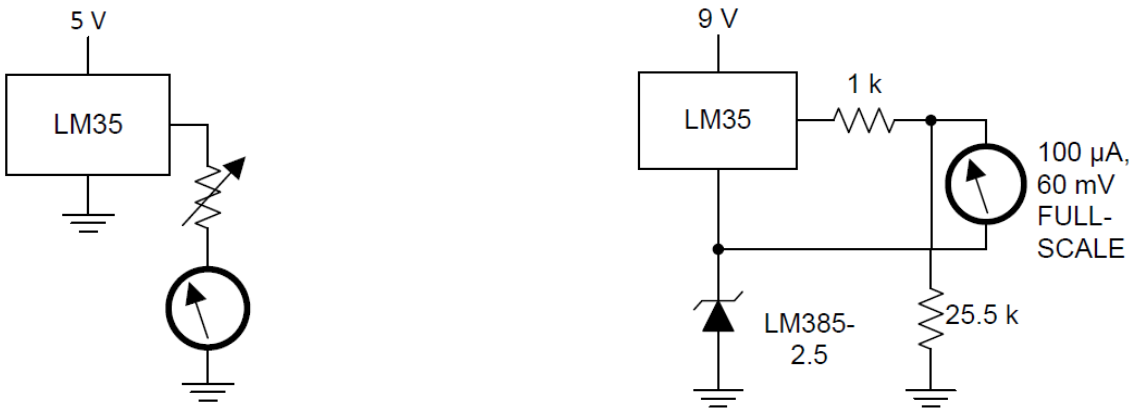


Рис.3.18. Термометр Фаренгейта



а) б)

Рис.3.19. Термометр Цельсія (аналоговий вимірювач) (а) і термометр Фаренгейта з розширеною шкалою (від 50 °F до 80 °F)

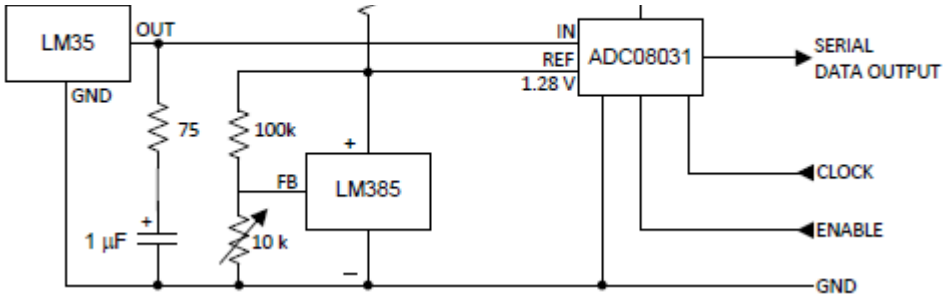


Рис.3.20. Схема перетворення температури в цифровий код (повна шкала 128 °C) з використанням АЦП ADC08031 з послідовним інтерфейсом

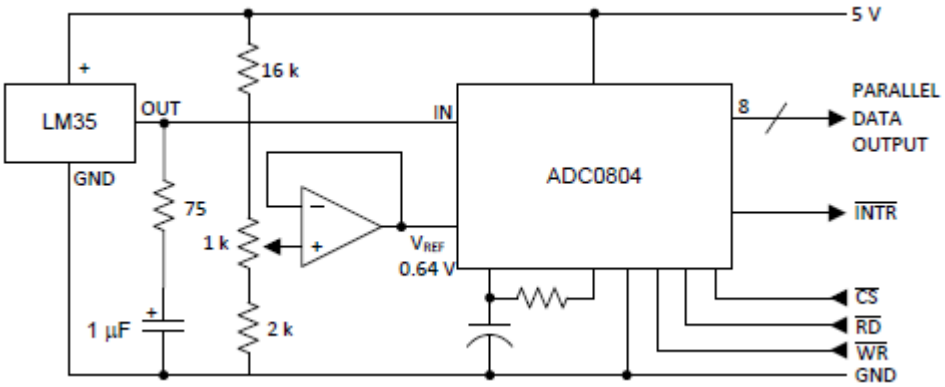


Рис.3.21. Схема перетворення температури в цифровий код з використанням АЦП ADC0804 з паралельним інтерфейсом (паралельні виходи TRI-STATE для стандартної шини даних на інтерфейс µP) (повна шкала 128 °C)

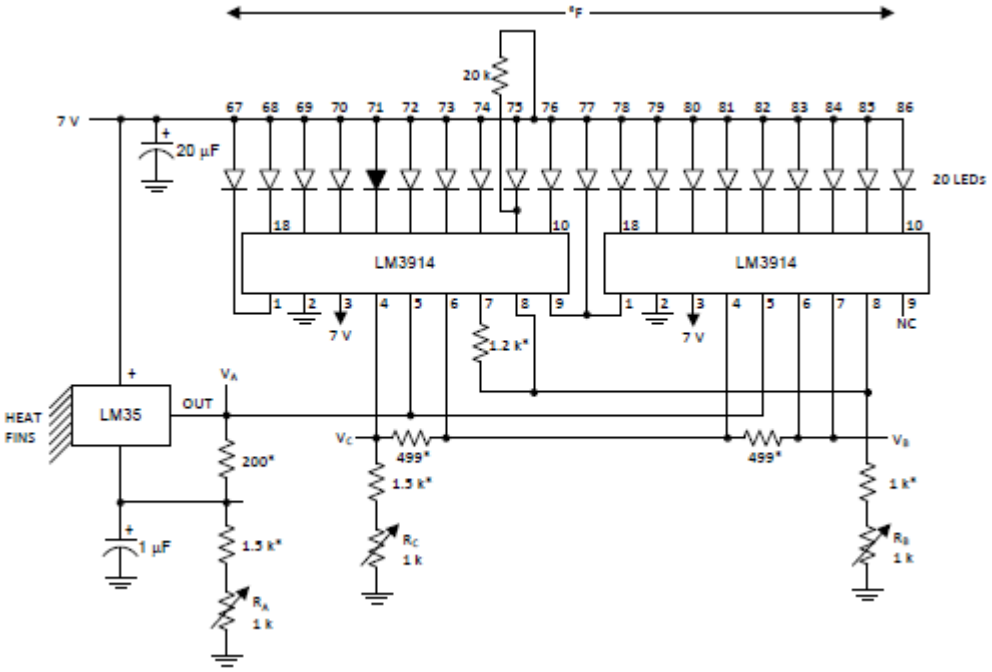


Рис.3.22. Схема виводу температури у вигляді гістограми (точковий режим)

*=1% або 2% тонкоплівковий резистор

R_B для V_B = 3,075 В

R_C для V_C = 1,955 В

R_A для V_A = 0,075 + 100 мВ / °C × $T_{ambient}$. Наприклад, V_A = 2,275 В при 22 °C

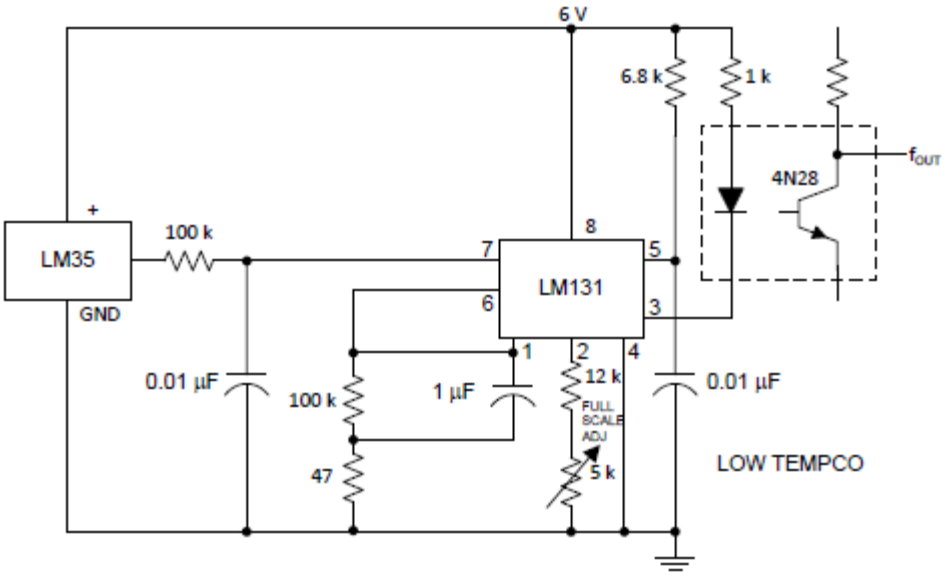


Рис.3.23. Схема з ізольованим виходом перетворювача напруги з давача LM35 в частоту (2 °C до 150 °C; 20 до 1500 Гц)

Мікросхема – драйвер для 8 – розрядних світлодіодних дисплеїв з послідовним інтерфейсом

При класичному управлінні семисегментним або матричними індикаторами з допомогою мікроконтролера виникає декілька проблем:

- збільшення кількості ліній управління;
- необхідність встановлювати потужні буферні елементи;
- необхідність постійної роботи мікроконтролера для підтримки динамічної індикації.

Як правило, всі ці проблеми вирішують методом роздування (розширення) схем і програм мікроконтролерів, але є спеціалізовані мікросхеми, що дозволяють вирішити ряд проблем при динамічній індикації. Однією з таких мікросхем є MAX7219 або MAX7221.

Загальні відомості про мікросхему MAX7219 або MAX7221

Драйвер MAX7219 керується (управляється) по трьох провідній послідовній шині Microwire (3-Wire). Мікросхема MAX7221 управляється по шині SPI і має обмежену швидкість наростання напруги драйверів сегментів для зменшення випромінювання EMI. Драйвери допускають каскадування для управління великою кількістю індикаторів. Кожний з розрядів індикатора має незалежну адресацію і його вміст може бути поновлений без необхідності перезапису всього індикатора. Мікросхема IC MAX7219/MAX7221 також дозволяє користувачу визначити режим декодування кожного розряда [11].

Крім того, драйвери MAX7219/MAX7221 мають режим сну з запам’ятовуванням інформації, аналогове і цифрове управління яскравістю підключених індикаторів і тестовий режим, який включає всі LED сегменти.
MAX7219 (MAX7221) – драйвер восьми розрядного цифрового LED індикатора з послідовним інтерфейсом. Драйвер може управляти вісьмома семи сегментними індикаторами з крапкою, або окремо (індивідуально) 64 світлодіодами в LED панелях з спільним катодом. Таким чином, ці мікросхеми підходять не тільки для семи сегментних, але і для матричних індикаторів.

Типова схема включення мікросхеми MAX7219

На рисунку 3.24 зображено типову схему включення мікросхеми MAX7219 [11].

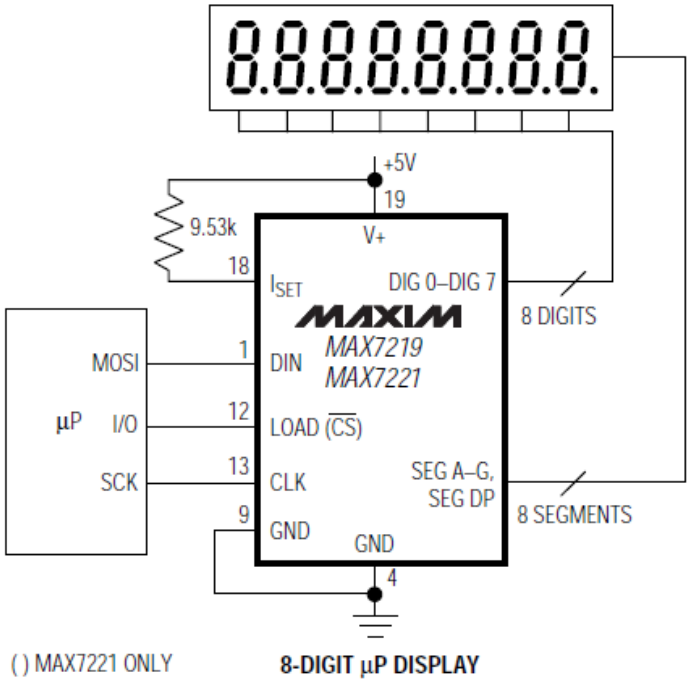


Рис.3.24. Схема включення мікросхеми MAX7219

З допомогою резистора на виводі 18 задається струм I_{SET}, яким встановлюється струм через світлодіоди, що дозволяє регулювати свічення сегментів індикатора “аналоговим” способом. В MAX7219/MAX7221 передбачене регулювання яскравості сегментів з допомогою ШІМа (PWM).

Розміщення виводів MAX7219

На рисунку 3.25 показано розміщення виводів драйвера – мікросхеми MAX7219/MAX7221.

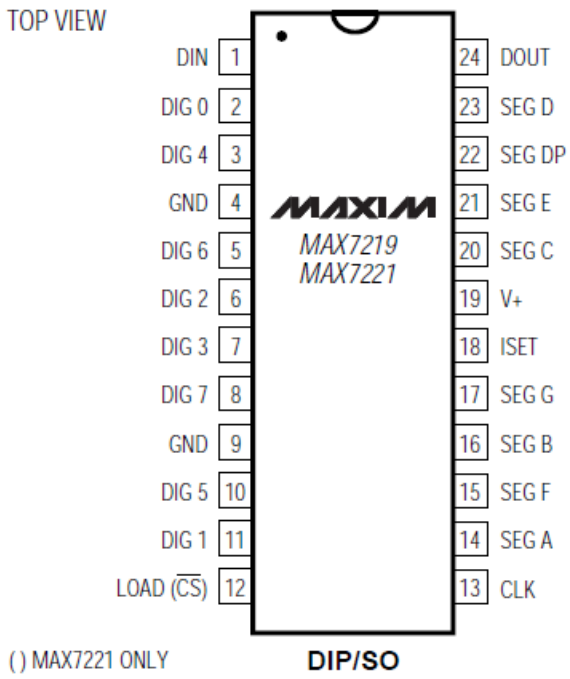


Рис.3.25. Розпіновка мікросхеми MAX7219

Таблиця 3.1

Призначення виводів мікросхеми MAX7219

Вивід	Назва (Найменування)	Функція
1	DIN	Вхід даних послідовно порта. Призначений для запису шістнадцятирозрядного слова по позитивному (додатному) фронту тактового сигналу (CLK). Запис відбувається старшим розрядом вперед. (Спочатку записується старший розряд)
2, 3, 5 - 8, 10, 11	DIG0 - DIG7	Низьким рівнем включає один з восьми індикаторів.
4, 9	GND	Мінусовий (спільний) вивід (провід) живлення
12	LOAD (для MAX7219)	(Додатнім) Позитивним фронтом цього імпульсу защіпаються останні 16 біт даних, що поступили (надійшли) на вхід DIN
	CS	Низький рівень дозволяє приймати дані по входу DIN, тобто даний сигнал відіграє роль вибору чіпа низьким рівнем. Позитивним фронтом дані, що надійшли, защіпаються
	CLK	Вхід тактування. Дані на вході DIN зчитуються при наростаючому (позитивному) фронті синхроімпульсів
14 – 17,	SEG A – SEG	Включають (вмикають) окремі (індивідуалні)

20 - 23	G, DP	сегменти позитивним імпульсом
18	ISET	Аналогове управління струмом через світлодіоди.
19	V+	Позитивний (додатний) вивід живлення (+5В)
24	DOUT	Вивід даних на наступні MAX7219 (MAX7221) для каскадного включення мікросхем MAX719 (MAX7221)

Основні характеристики мікросхеми MAX7219:

- Частота тактування інтерфейсу SPI: 10 МГц
- Напруга живлення: 4,5…5,5 В
- Споживання по шині +5В в режимі сну: 150 мкА
- Струм через один сегмент в імпульсі: до 320 мА
- Середній струм через один сегмент: до 40 мА
- Частота слідування імпульсів включення символів: не менше 500 Гц
- Затримка. Поступлення даних –> вивід на дисплей: 2,2 мс.

Застосування мікросхеми MAX7219

Формат даних, що передаються в MAX7219 показано в таблиці 3.2. Дані відсилаються по 16 біт, старшим бітом вперед. Спочатку відсилається старший біт.

Таблиця 3.2

Формат послідовних даних (16 біт) MAX7219

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB (ст. біт) DATA LSB (мол. біт)							

В бітах D15…D12 корисної інформації не передається. Як правило, в цих бітах передаються нулі. В полі ADDRESS (адреса) вказується дія, яку потрібно виконати. Якщо в ADDRESS передається значення 1…8 (0001…1000), то це вибір знакомісця. В полі DATA (дані) в цьому випадку пересилається інформація про сегменти вибраного знакомісця. Якщо DATA рівне 0 (00000000), то сегменти не світяться. Якщо DATA рівне 255 (11111111), то світяться всі сегменти вибраного знакомісця, включаючи крапку. В режимі декодування DATA несе іншу інформацію, детальніше в datasheet на

MAX7219/7221. Якщо в ADDRESS передається значення 9…15 (1001…1111), то ця вказівка виконає службову інструкцію. В таблиці 3.3 наведено значення бітів команд ADDRESS D8…D11 та їх опис.

Таблиця 3.3

Команди MAX7219

Адреса					Опис команди
D11	D10	D9	D8	Hex	
1	0	0	1	0x09	Режим декодування. Бітами DATA вибирається які знакомісця декодувати, які ні. Dx =1 – декодувати знакомісце x, Dx=0 – не декодовувати знакомісце x.
1	0	1	0	0x0A	Інтенсивність свічення (яскравість). Бітами D0…D3 вибирається яскравість свічення. При D0=D1=D2=D3=0 яскравість мінімальна. При D0=D1=D2=D3=1 яскравість максимальна.
1	0	1	1	0x0B	Вибір кількості відображувальних знакомісць, що будуть відображені. Бітами D0..D2 вибираються знакомісця, що відображаються. При D0=D1=D2=1 відображаються всі вісім знакомісць.
1	1	0	0	0x0C	Режим сну. При DATA=0 мікросхема впадає (переходить) в режим сну. DATA=1 являється нормальним режимом.
1	1	0	1	0x0D	Не використовується
1	1	1	0	0x0E	Не використовується
1	1	1	1	0x0F	Тест. Якщо DATA=1 – тест включено, якщо DATA=0 – виключений.

Для нормальної роботи мікросхеми її необхідно ініціалізувати після подачі живлення. Ініціалізація полягє у виконанні дякої послідовності команд, після якої мікросхема починає нормально працювати і реагувати на команди і дані. Без ініціалізації мікросхема нічого не буде висвічувати, але тим не менше, при цьому мікросхема спокійно переходить в режим TEST (ADDRESS = 0x0F) і засвічує всі сегменти одночасно.

Для ініціалізації потрібно виконати:

1. ADDRESS = 0x0F, DATA = 0x00 – Тест індикатор включений
2. ADDRESS = 0x0C, DATA = 0x01 – Вийти зі сну
3. ADDRESS = 0x0B, DATA = 0x07 – Кількість задіяних символів — 8 символів
4. ADDRESS = 0x09, DATA = 0x00 – Деширфтори відключені

5. ADDRESS = 0x0A, DATA = 0x0F – Інтенсивність свічення (яскравість) максимальна

Після цього на сегменти дисплею будуть світитися випадковим чином. Це потрібно враховувати і після ініціалізації виконати очистку дисплея, наприклад, записати у всі знакомісця символ 0 або погасити всі сегменти, наприклад, використовуючи ADDRESS = 0x01…0x08, DATA=0x00.

MAX7219 – це компактний драйвер дисплею з послідовним вводом-виводом і спільним катодом (компактний драйвер з послідовним вводом-виводом для дисплею з спільним катодом), який зв'язує мікропроцесори (uP), мікроконтролери з 7 – сегментними цифровими світлодіодними дисплеями, що відображають до 8 цифр, гістограмними дисплеями, або 64 окремими світло діодами (рисунок 3.26). Мікросхема містить декодер двійково-десяткового коду B, схема мультиплексорного сканування, драйвера сегментів і цифр, а також статичне ОЗП 8x8, в якому зберігається кожна цифра. Для встановлення струму сегмента для всіх світло діодів потрібно тільки один зовнішній резистор. Мікросхема-драйвер MAX7221 сумісна з SPI, OSPI і Microwire і має сегментні драйвери з обмеженням шивлкості наростання для зменшення електромагнітних перешкод. Зручний 3 – провідний послідовний інтерфейс дає змогу підключитися до всіх поширених мікроконтролерів, мікропроцесорів. MAX7219/MAX7221 також дозволяє користувачу вибрати декодування з кодом B або без декодування для кожної цифри.

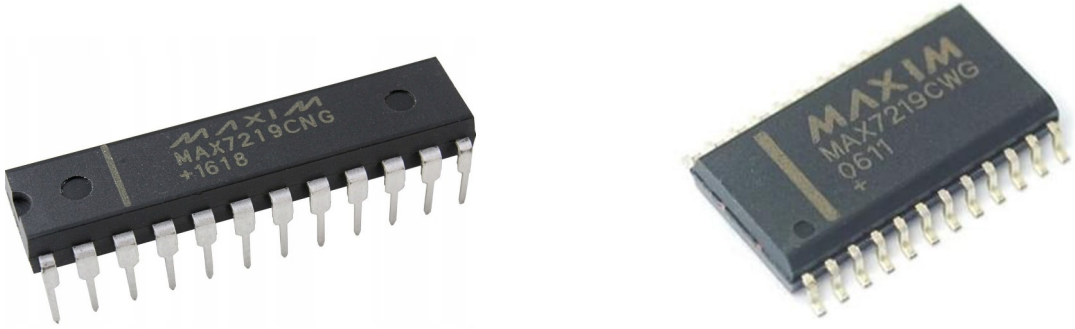


Рис.3.26. Мікросхема – драйвер з послідовним інтерфейсом для управління LED дисплеєм зі спільним катодом

Пристрій включає в собі режим відключення з низьким енергоспоживанням 150 мкА, аналогове і цифрове управління яскравістю,

регістр обмеження сканування, що дозволяє користувачу відображати від 1 до 8 цифр, і тестовий режим, який примусово засвічує всі світлодіоди [12].

Мікросхема може бути використана для управління гістограмним дисплеєм, 7 сегментним дисплеєм, промисловим контролером, панельним лічильником, світлодіодним матричним дисплеєм (LED matrix displays).

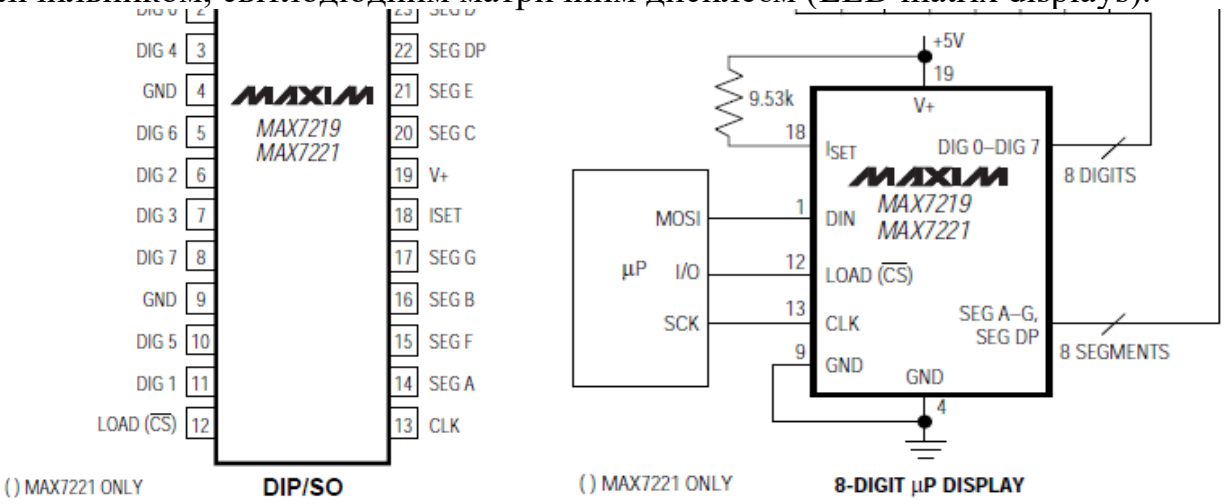


Рис.3.27. Розміщення та призначення виводів мікросхеми MAX7219 і типова схема включення

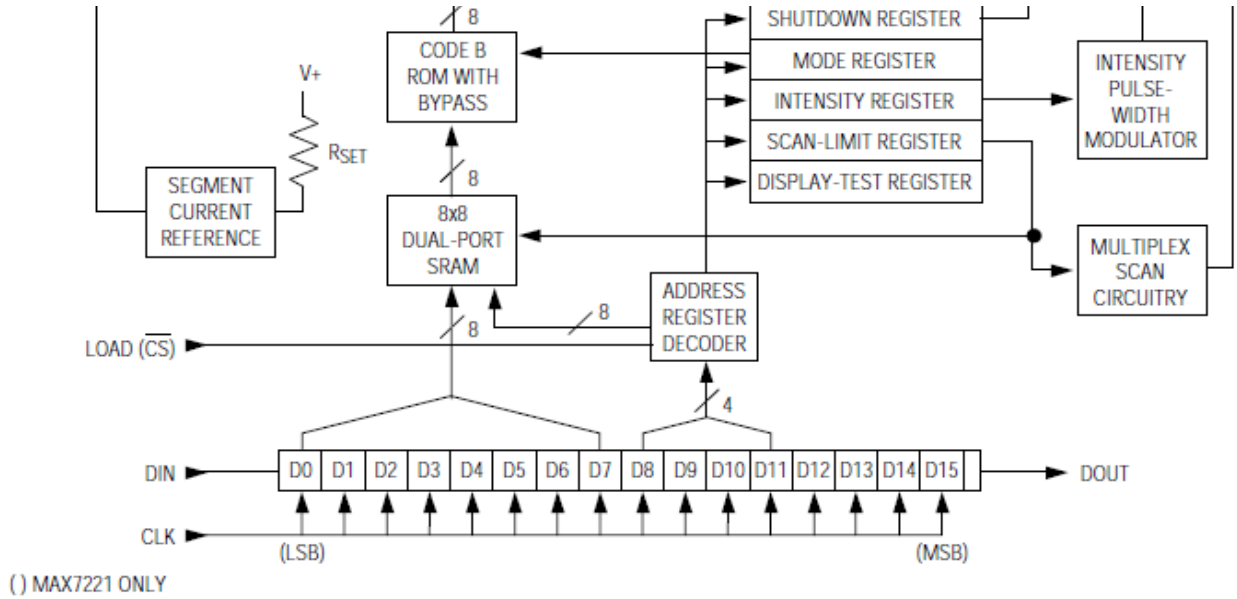


Рис.3.28. Функціональна діаграма MAX7219/MAX7221

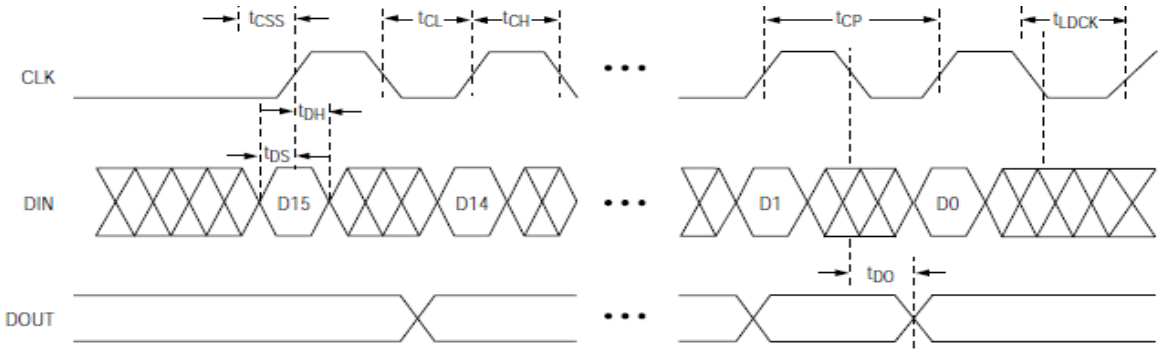


Рис.3.29. Часова діаграма MAX7219/MAX7221

Таблиця 3.4

Карта адрес регістрів

Регістр	D15-D12	D11	D10	D9	D8	Hex код
No-Op	X	0	0	0		X0
Цифра 0	X	0	0	0		X1
Цифра 1	X	0	0	1		X2
Цифра 2	X	0	0	1		X3
Цифра 3	X	0	1	0		X4
Цифра 4	X	0	1	0		X5
Цифра 5	X	0	1			X6
Цифра 6	X	0	1			X7
Цифра 7	X	1	1			X8
Режис декодування	X	1	0			X9
Інтенсивність	X	1	0			XA
Ліміт (границя) сканування (розгортки)	X	1	0			XB
Відключення	X	1	0			XC
Тест дисплею	X	1	1			XF

Таблиця 3.5

Формат регістра відключення (адреса (Hex) = XC)

Режим	Адресний код (Hex)	Дані регістра							
		D7	D6	D5	D4	D3	D2	D1	D0
Режим відключення	XC	X	X	X	X	X	X	X	0
Нормальний режим	XC	X	X	X	X	X	X	X	1

Таблиця 3.6

Приклади регістра режимів декодування (адреса (Hex) = X9)

Режим декодування	Дані регістра								Hex код
	D7	D6	D5	D4	D3	D2	D1	D0	
Немає декодування для цифр 7-0	0	0	0	0	0	0	0	0	00
Декодуваня коду B для цифри 0. Немає декодування для цифр 7-1	0	0	0	0	0	0	0	1	01
Декодуванн я коду B для цифр 3-0. Немає декодування для цифр 7-4	0	0	0	0	1	1	1	1	0F
Декодуванн я для цифр 7-0	1	1	1	1	1	1	1	1	FF

Таблиця3.7

Фонт коду B

7 – сегментний символ	Дані регістра							Сегменти Вкл. = 1						
	D7*	D6-D4	D3	D2	D1	D0	DP	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
-		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

* десяткова крапка встановлюється бітом D7 = 1

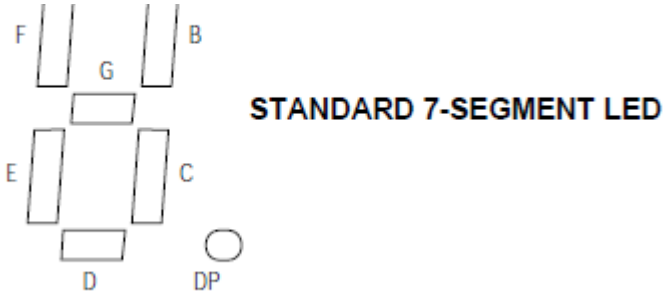


Рис.3.30. Стандартний 7 – сегментний LED (світлодіодний) індикатор

Таблиця 3.8

Регістр даних і відповідні сегментні лінії

	Дані регістра							
	D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Відповідна сегментна лінія	DP	A	B	C	D	E	F	G

Таблиця 3.9

Формат регістра інтенсивності (адреса (Hex) = XA)

Робочий цикл MAX7219	D7	D6	D5	D4	D3	D2	D1	D0	Hex – код
1/32 (мін. on)	X	X	X	X	0	0	0	0	X0
3/32	X	X	X	X	0	0	0	1	X1
5/32	X	X	X	X	0	0	1	0	X2
7/32	X	X	X	X	0	0	1	1	X3
9/32	X	X	X	X	0	1	0	0	X4
11/32	X	X	X	X	0	1	0	1	X5
13/32	X	X	X	X	0	1	1	0	X6
15/32	X	X	X	X	0	1	1	1	X7
17/32	X	X	X	X	1	0	0	0	X8
19/32	X	X	X	X	1	0	0	1	X9
21/32	X	X	X	X	1	0	1	0	XA
23/32	X	X	X	X	1	0	1	1	XB
25/32	X	X	X	X	1	1	0	0	XC
27/32	X	X	X	X	1	1	0	1	XD
29/32	X	X	X	X	1	1	1	0	XE
31/32	X	X	X	X	1	1	1	1	XF

Таблиця 3.10

Формат регістра границі сканування (адреса (Hex - шістнадцяткове) = XB)

Ліміт сканування	Дані регістра								Hex код
	D7	D6	D5	D4	D3	D2	D1	D0	
Відображення цифри 0 тільки	X	X	X	X	X	0	0	0	X0
Відображення цифр 0 і 1	X	X	X	X	X	0	0	1	X1
Відображення цифр 0 1 2	X	X	X	X	X	0	1	0	X2
Відображення цифр 0 1 2 3	X	X	X	X	X	0	1	1	X3
Відображення цифр 0 1 2 3 4	X	X	X	X	X	1	0	0	X4
Відображення цифр 0 1 2 3 4 5	X	X	X	X	X	1	0	1	X5
Відображення цифр 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	X6
Відображення цифр 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	X7

Таблиця 3.11

Максимальний струм сегмента для 1-, 2-, або 3-х цифрових (розрядних) дисплеїв

Кількість (число задіяних) цифр для відображення	Максимальний струм сегмента (мА)
1	10
2	20
3	30

Табл иця 3.12

Формат регістра тестування дисплею (адреса (Hex) = XF)

Режим	Дані регістра							
	D7	D6	D5	D4	D3	D2	D1	D0
Нормальний (звичайний) режим роботи	X	X	X	X	X	X	X	0
Режим тесту дисплея	X	X	X	X	X	X	X	1

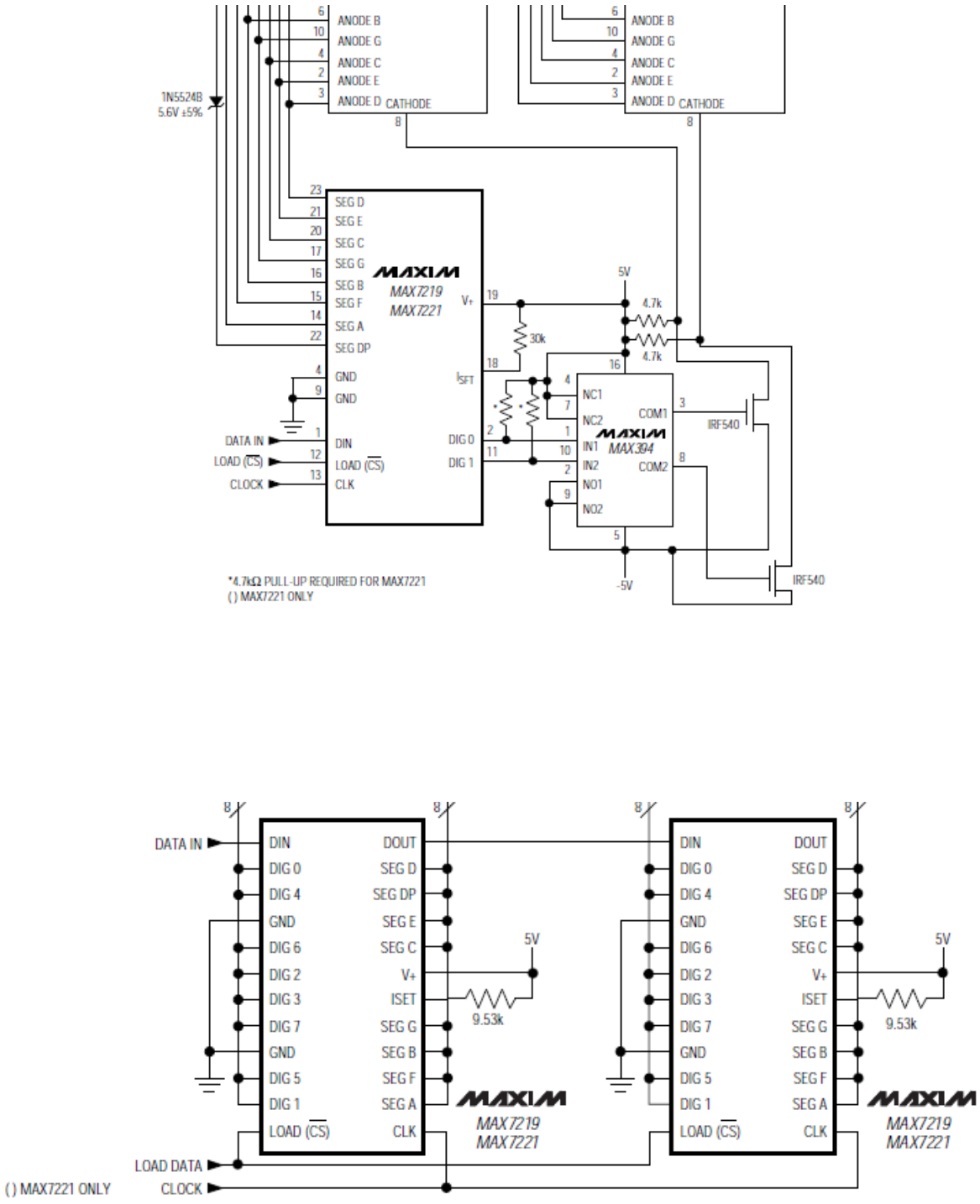


Рис.3.31. Підключення мікросхеми MAX7129 для управління дисплеєм з 7 сегментних індикаторів

Характеристики мікросхеми – драйвера MAX7219:

- послідовний інтерфейс 10 МГц;
- індивідуальне управління сегментами світло діодів;

- вибір цифри декодування/недокудування;
- відключення при низькому енергоспоживанні 150 мкА (дані зберігаються);
- цифрове і аналогове управління яскравістю;
- дисплей гасне при включенні живлення;
- світлодіодний дисплей з спільним катодом управління;
- драйвери сегмента з обмеженою швидкістю наростання для зменшення електромагнітних перешкод (MAX7221);
- SPI, QSPI, послідовний інтерфейс Microwire (MAX7221);
- 24 – контактний (вивідний) корпус DIP і SO.

Таблиця 3.12

Призначення виводів мікросхеми MAX7219

Пін	Назва	Функція
1		Послідовний ввід даних. Дані завантажуються у внітршній 16 – бітний регстр зсуву по передньому фронту CLK.
2, 3. 5 – 8, 10, 11	DIG 0 - DIG 7	Восьмирозрядні лінії управління, який приймають струм від спільного катоду дисплея. Мікросхема MAX7219 підтягує цифрові виходи до V+ при виключенні (відключенні). Драйвери мікросхеми MAX7221 мають високий опір при виключеному стані.
4, 9	GND	Ground (GND). Земля (обидва виводі GND мають бути підключені)
12	LOAD (MAX7219)	Вхід завантаження даних. Останні 16 біт послідовних даних фіксуються по передньому (наростаючому) фронту сигнала LOAD.
	CS (MAX7221)	Вибір чіпа (мікросхеми). Послідовні дані завантажуються в регістр зсуву поки на виводі CS низькій логічний рівень. Останні 16 біт послідовних даних фіксуються по передньому (наростаючому) фронту CS.
13	CLK	Послідовний вхід тактових сигналів (вхід синхронізації). Максимальна швидкість 10 МГц. По передньому (наростаючому) фронту сигналу CLK дані зсуваються у внутрішній регістр зсуву. По задньому (спадаючому) фронту CLK дані тактуються з DOUT. На мікросхемі MAX7221 вхід CLK активний тільки при низькому рівні CS.
14 –	SEG A –	Семисегментні драйвери і драйвер десяткової коми

17, 20 - 23	SEG G, G, DP	(крапки), які подають струм на дисплей. На мікросхемі MAX7219, коли драйвер сегмента виключений, він підтягується до GND. Драйвери сегментів MAX7221 мають високий опір (імпеданс) у виключеному стані.
18	ISET	Підключається до VDD через резистор (RSET), щоб встановити піковий струм сегмента.
19	V+	Додатня напруга живлення. Підключається до +5В.
24	DOUT	Вивід послідовних даних. Дані в DIN дійсні на DOUT через 16,5 тактів. Цей вивід використовується для послідовного підключення декількох мікросхем MAX7219/MAX7221 і ніколи не має високого імпедансу.

Управління матричним світлодіодним дисплем з допомогою мікросхеми MAX7219 і плати Arduino [11, 12]. Використання точкового матричного дисплею 8x8 і 8x32 для відображення тексту, символів і прокручування тексту. Модуль світлодіодної матриці 8x32 зі світлодіодним драйвером MAX7219 сумісний з платою Arduino та іншими мікроконтролерами. Світлодіодна матриця 8x32 має 256 світлодіодів, які роміщені у вигляді матриці з 8 рядків та 32 стовпців. Відповідно, він називається світлодіодною матрицею 8x32. З допомогою бібліотек Parola і MAX72xx можна генерувати різні рухомі світлодіодні шаблони, такі як числа, алфавіти, біжучий (прокручувальний) текст, символи, смайлики і т.д. Крім того, на дисплеї можна відображати дані з давачів.

Світлодіодний дисплей з точковою матрицею

На сьогоднішній день на ринку є доступними світлодіодні матриці з різним кольором свічення і розміром матриці. Є матриці одноколірні, двоколірні, багатоколірні і світлодіодні матриці RGB. Вони також доступні в різних розмірах 5x7, 8x8, 16x16, 8x32, 32x32 і т.д. (рисунок 3.32).

Рис.3.32. Світлодіодний дисплей з точковою матрицею

Світлодіодний матричний дисплей 8x32 представляє собою кластер з 4 окремих модулів, які з'єднані всередині (рисунок 3.33). Ці модулі також можуть бути розділені, оскільки кожний модуль містить один і той самий чіп Maxim MAX7219 і має однакове живлення і підключення для передачі даних.

Рис.3.33. Світлодіодний дисплей з точковими матрицями

Модуль матричного дисплею 4 в 1 складаєтьмя з чотирьох 8x8 зелених точкових матриць зі спільним катодом. Кожна матриця має мікросхему – драйвер управління MAX7219. Модуль ідеально підходить для відображення біжучого тексту і зображення. Модуль можна каскадувати на більш великий матричний дисплей, але для цього потрібно забезпечити його достатнім струмом від 5 В блока живлення.

Характеристики дисплейного модуля з світлодіодних точкових матриць і драйвером MAX7219:

- каскадовані чотири точкові матриці зеленого (червоного) свічення зі спільним катодом;
- робоча напруга живлення світлодіодів: 5 В;
- чотири отвори для кріпильних гвинтів в кожній точковій матриці;
- всього 16 отворів, діаметр отвору 3 мм;
- модуль з вхідними і вихідними інтерфейсами;
- підтримка каскадування декількох модулів;
- розмір: 12,8 x 3,2 x 1,3 см (Д*Ш*В).

Розміщення та призначення виводів

Світлодіодна матриця 8х8 має 8 позитивних і 8 негативних виводів (контактів). Вісім негативних виводів (контактів) – це 8 стовпців, а вісім позитивних виводів (контактів) – 8 рядків (рисунок 3.34).

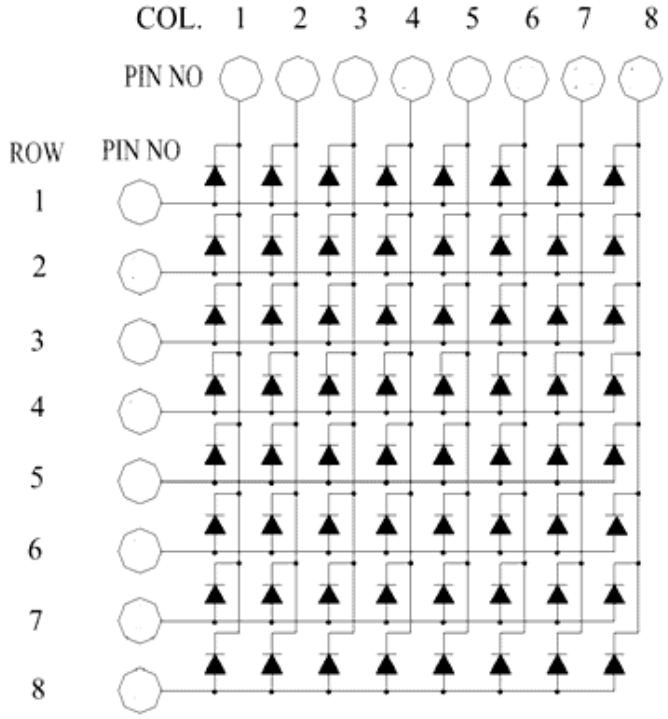


Рис.3.34. Світлодіодна матриця

Чотири світлодіодних матриці 8х8 з’єднані між собою через виводи мікросхеми MAX7129 (рисунок 3.35).

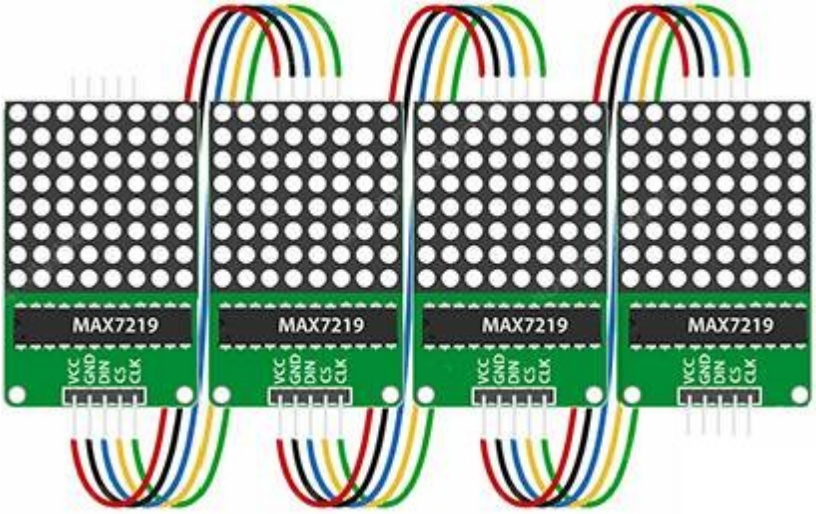


Рис.3.35. З’єднання чотирьох світлодіодних матриць 8х8

Мікросхема – драйвер світлодіодів MAX719

Світлодіодна матриця може управлятися двома способами. Перший – це паралельний, коли кожний рядок або стовпець адресуються паралельними даними, а другий – послідовний, коли дані відсилаються послідовно, а мікросхема-драйвер перетворює ці послідовні дані в паралельні.

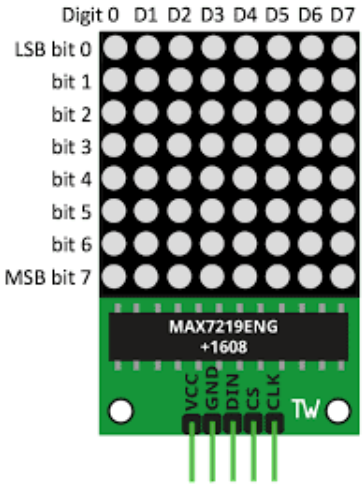


Рис.3.36. Світлодіодна матриця

Мікросхема MAX7219 – це драйвер з послідовним входом і паралельним виходом для управління дисплем зі спільним катодом. Він використовується для підключення мікропроцесорів і мікроконтролерів з 64 індивідуальними світлодіодами. Світлодіодна матриця 8x8 підключена до MAX7219. Вхідні дані надходять з плати Arduino до мікросхеми MAX7219.

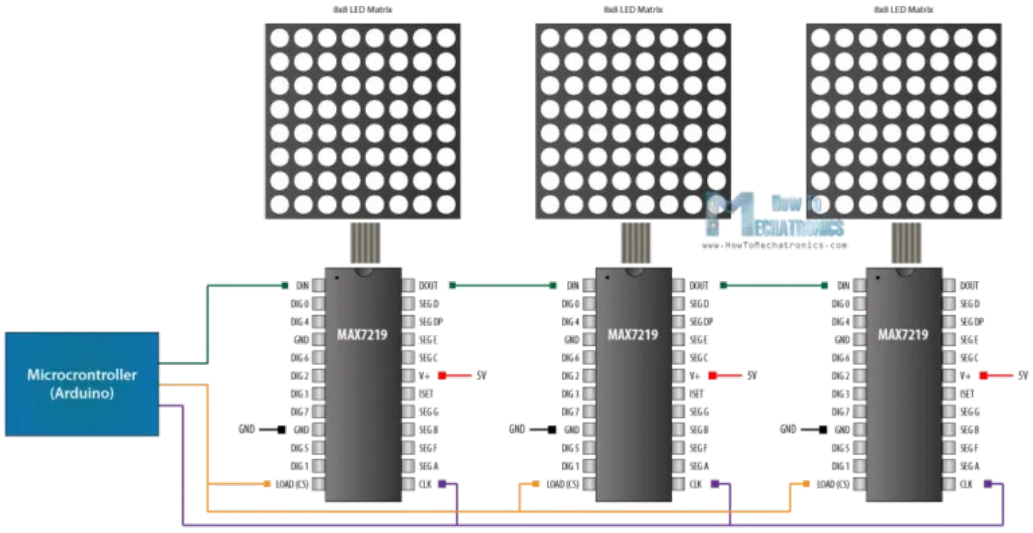


Рис.3.37. Послідовне підключення декількох світлодіодних матриць з мікросхемами-драйверами MAX7219

Мікросхема – драйвер MAX7219 може управляти 64 індивідуальними світлодіодами використовуючи тільки 3 проводи для комунікації з Arduino. Більщ того, можна послідовно підключити декілька драйверів і матриць з використанням тих самих 3 проводів.

64 світлодіода управляються 16 вихідними пінами мікросхеми. Максимальна кількість світлододів, які засвічуються одночасно, насправді вісім (8). Світлодіоди розміщені у вигляді набору рядків і стовпців 8x8. Таким чином, MAX7219 активує кожний стовпець на дуже короткий період часу і в той же час управляє кожним рядком. Швидко переключаючись між стовпцями і рядками, людське око замітить тільки неперервний текст.

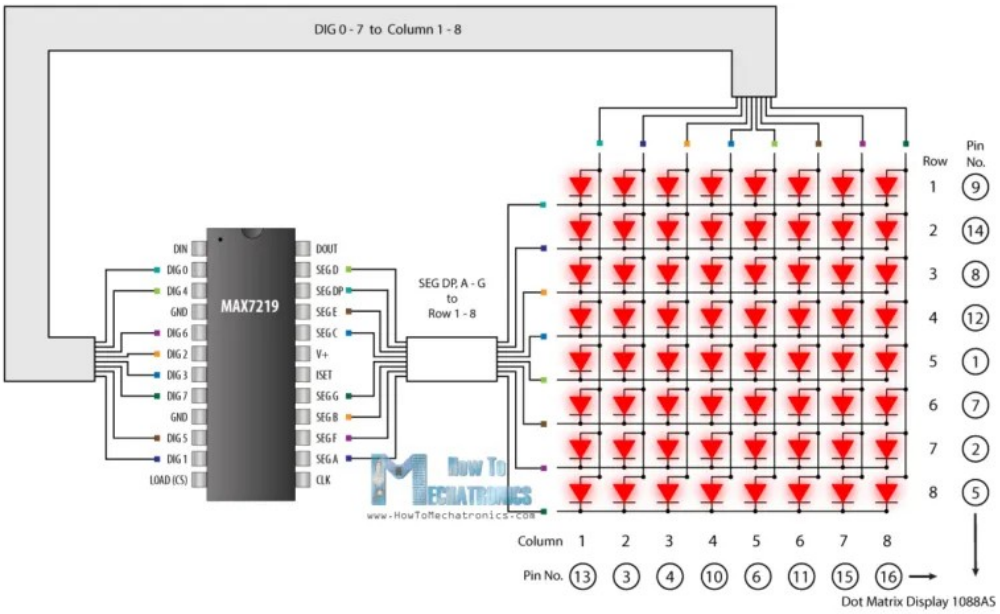


Рис.3.38. Підключення світлодіодної матриці до мікросхеми-драйвера MAX7219

Слід звернути увагу на внутрішнє розміщення виводів звичайної світлодіодної матриці 8x8, тому, якщо потрібно зібрати матрицю самостійно, то це потріюно враховувати. Також звичайна комутаційна плата для MAX7219 постачається з резистором між 5V і 18 виводом мікросхеми. Резистор використовується для налаштування яскравості або струму, який протікає через світлодіоди.

В таблиці 3.13 з datasheet на мікросхему MAX7219 показано значення резистора, яке потрібно використовувати у відповідності з прямим падінням напруги на світлодіодах.

Таблиця 3.13

R_{SET} vs струм сегмента і пряме падіння напруги на світлодіоді

I_{SEG} (мА)	**V_{LED} (В)**				
	1,5	**2,0**	**2,5**	**3,0**	**3,5**
40	12,2	11,8	11,0	10,6	9,69
30	17,8	17,1	15,8	15,0	14,0
20	29,8	28,0	25,9	24,5	22,6
10	66,7	63,7	59,3	55,4	51,2

Мультиплексування світлодіодної матриці

Для управління 8x8, тобто 64 світлодіодами, потрібно 64 виводи даних. Але з допомогою мультиплексування світлодіодів можна управляти такою ж кількістю світлодіодів, використовуючи 16 виводів даних. В цьому процесі всі катоди з'єднані разом порядково, а всі аноди з'єднані разом по стовпцях.

На рисунку 3.39 показано принципову схему мультиплексування світлодіодної матриці 8x8.

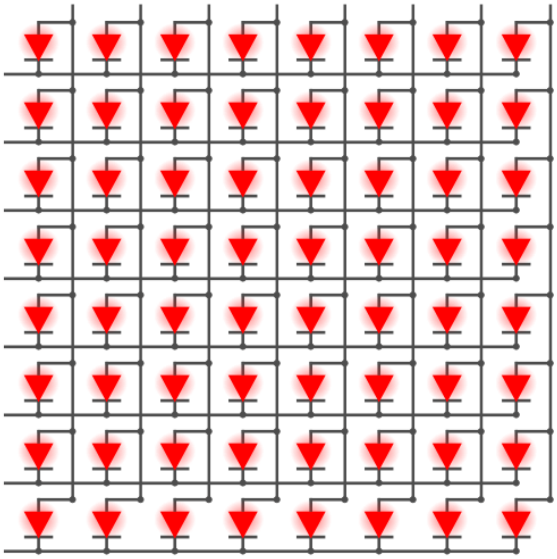


Рис.3.39. Принципова схема мультиплексування світлодіодів матриці

В процесі мультиплексування активують кожний стовпець на дуже короткий проміжок часу, що засвічує необхідний світлодіод для цього рядка. Такий же процес виконують для кожного стовпця. Стовпці переключаються

так швидко, що звичайне людське око бачить це як цілісну картину. Для кращого розуміння, наприклад, можна створити анімацію, яка показує, як буква ‘A’ виводиться на світлодіодному матричному дисплеї 8x8.

Отже, для управління світлодіодною матрицею 8x8 при мультиплексуванні потрібно 16 виводів (контактів) даних. Для ще більшого зменшення кількості виводів можна використати мікросхему – драйвер світлодіодів MAX7219. Світлодіодний драйвер MAX7219 є дуже популярним драйвером для світлодіодних матричних дисплеїв 8x8. На рисунку 3.40 показано розпіновку мікросхеми MAX7219.

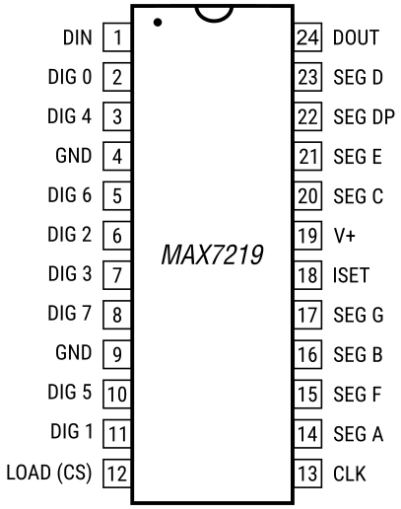


Рис.3.40. Розміщення виводів на мікросхемі – драйвері MAX7219

Вона може бути виготовлена в 24 – вивідному корпусі DIP або SO. Мікросхема має 16 паралельних виводів даних для управління світлодіодами. Вона має інтерфейс SPI, що дозволяє підключити декілька модулів MAX7219 на одну і ту саму шину. Мікросхема приймає послідовні дані від мікроконтролера та перетворює їх в паралельні дані для управління світлодіодами. Тому, потрібно лише три виводи від мікроконтролера для управління світлодіодним матричним дисплеєм 8x8 з драйвером MAX7219.

Щоб униукнути складної проводки використовують світлодіодну матрицю з комутаційною платою MAX7219. Модуль Generic 8x8 і модуль FC-16 8x32 мають однакову конфігурацію виводів. Вони мають по п’ять штирьових виводів з кожної сторони. Одна сторона призначена для входу, а інша сторона для виводу. На вході він може отримувати команди від

мікроконтролерів, а на виході він може підключатися до іншого модуля дисплею. На вході є VCC, GND, DIN, CS і CLK, а на виході VCC, GND, DOUT, CS і CLK.

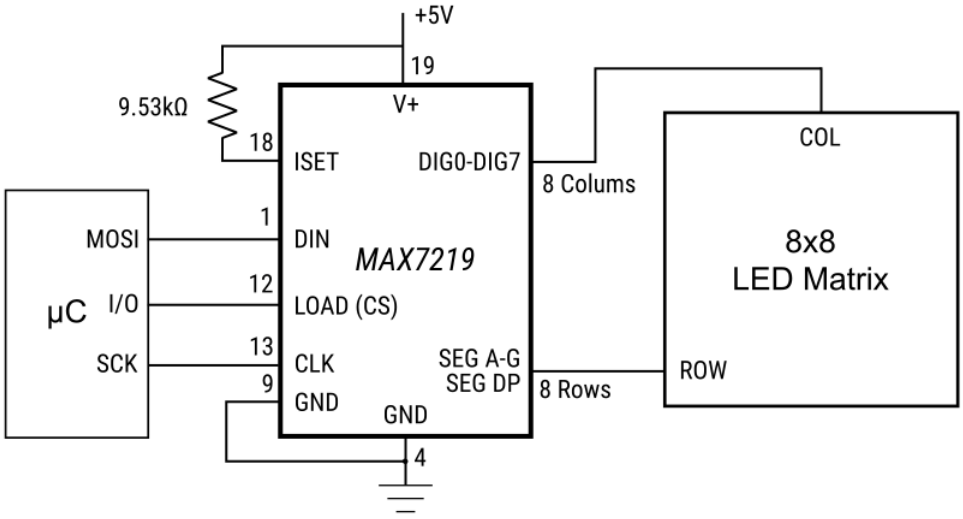


Рис.3.41. Схема підключення мікросхеми – драйвера MAX7219 до мікроконтролера і світлодіодного матричного дисплею 8x8

На рис.3.42 зображено розміщення виводів універсального модуля і модуля FC-16.

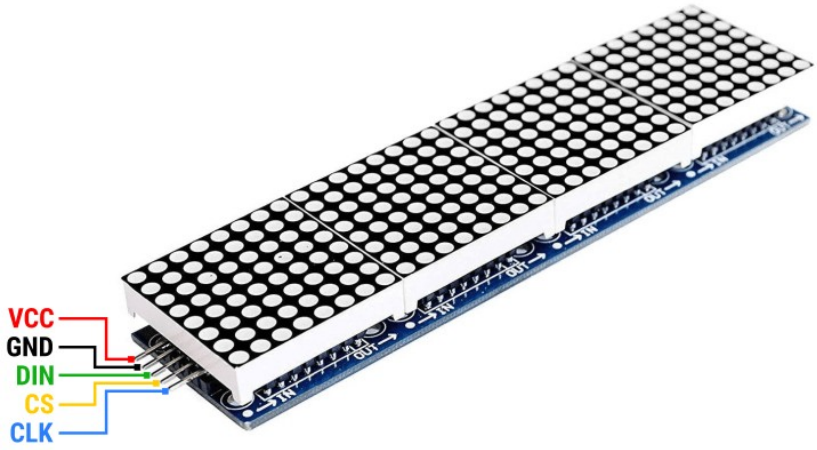


Рис.3.42. Розміщення виводів дисплейного модуля FC-16

Найбільш розповсюджені розміри світлодіодної матриці MAX7219 – 8x8 і 8x32. Світлодіодний матричний дисплей MAX7219 має п’ять виводів (контактів) VCC, GND, DIN, CS і CLK. Він використовує протокол зв’язку SPI. Таким чином, модуль можна підключити до плати Arduino, як і будь-який інший модуль SPI.

В цьому проекті використано плату Arduino Micro для управління модулем світлодіодних точкових матриць 8x32 на базі мікросхеми – драйвера MAX7219. Плата Arduino Micro має виділені виводи (апаратний SPI) для комунікації по SPI, але також для комунікації можна використовувати будь-які три довільних цифрових вивода (піни) (програмний SPI). Варто пам’ятати, що апаратний SPI набагато швидший за програмний SPI. На Рис. показано підключення плати Arduino Uno до дисплейного модуля FC-16 з допомогою апаратних виводів інтерфейса SPI.

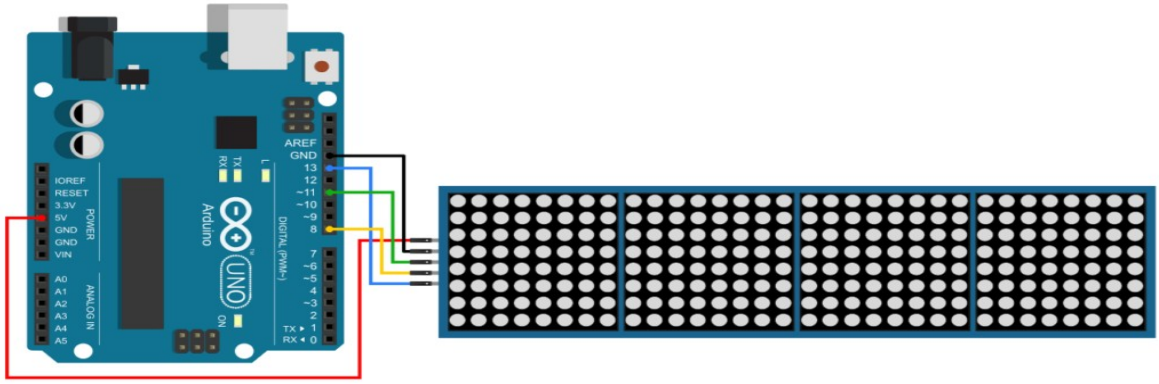


Рис.3.43. Підключення плати Arduino до дисплейного модуля з світлодіодних точкових матриць

Вивід VCC дисплейного модуля підключено до виводу Arduino 5V, а вивід GND до виводу землі плати Arduino. Вивід DIN модуля підключено до виводу 11 плати Arduino, який є виводом MOSI, і вивід CLK до виводу 13 Arduino, який є виводом SCK. Пін (вивід) вибору мікросхеми CS підключено до виводу 8 плати Arduino.

Взаємодія матричного світлодіодного дисплею 8x32 MAX7219 з платою Arduino. Розглянемо, як можна зв’язати точковий матричний світлодіодний дисплей 8x32 MAX7219 з платою Arduino для відображення тексту і чисел.

Цей дисплей споживає багато струму, тому модуль потрібно запускати (живити) від зовнішнього джерела живлення замість живлення 5 В від плати Arduino. Потрібно використовувати зовнішній адаптер живлення 5 В, 3 А, який підключений до виводів 5 В і GND на платі Arduino.

Оскільки модуль MAX7219 потребує передачі великої кількості даних, то його підключаємо до апаратних виводів інтерфейса SPI плати Arduino. Для плат Arduino Uno/Nano такими виводами є цифрові піни 13 (SCK), 12 (MISO), 11 (MOSI) і 10 (SS). Виводи CLK, CS, DIN матричного світлодіодного дисплею 8x32 підключені (підключаємо) до цифрових пінів 13, 10, 11 плати Arduino.

Якщо потрібно послідовно з'єднати декілька дисплеїв для створення дисплею великого розміру, то з'єднують вивід DOUT першого дисплею з виводом DIN наступного дисплею. Для послідовного з'єднання декількох дисплеїв і створення дисплею великого розміру потрібно з'єднати вивід DOUT першого дисплею з виводом DIN наступного дисплею. Виводи VCC, GND, CLK, CS будуть спільними для дисплеїв.

3.2. Проєктування цифрового годинника-термометра в САПР Proteus VSM

Розроблювальний цифровий пристрій повинен мати функції годинника і термометра. Пристрій має виводити поточний час і дату, значення температури в градусах Цельсія або Фаренгейта на матричний світлодіодний дисплей, також дозволяє налаштувати час і дату годинника реального часу DS1307.

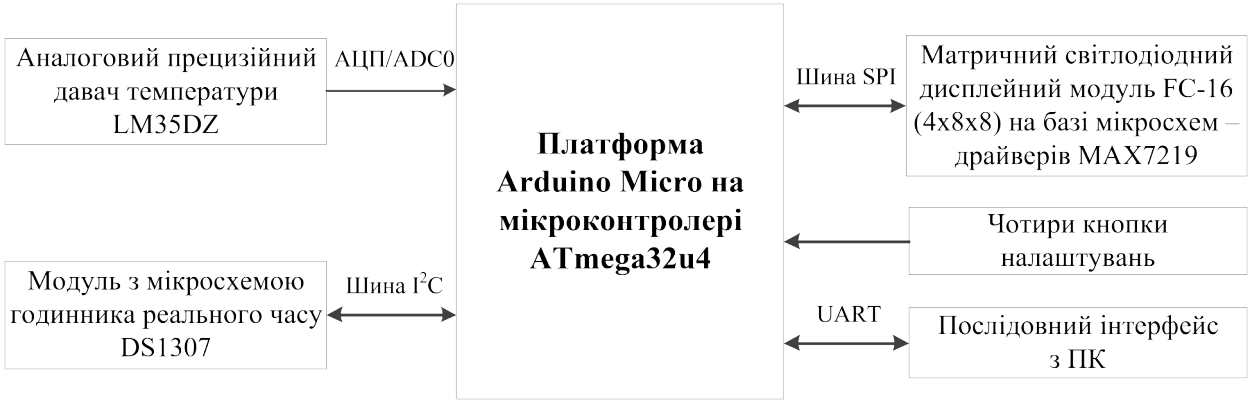


Рис.3.44. Структура цифрового годинника – термометра

На рис. 3.44 зображено запропоновану структуру цифрового годинника-термометра. Цифровий годинник-термометр включає в себе як апаратне так і програмне забезпечення. Апаратне забезпечення цифрового годинника -

теромеметра складається з плати Arduino Micro на мікро контролері ATmega32u4, який є головним модулем опрацювання даних, до якої підключено мікросхему годинника реального часу DS1307, аналоговий давач температури LM35DZ, матричний світлодіодний дисплейний модуль 4x8x8 FC-16 на базі мікросхем – драйверів управління MAX7219 і 4 кнопки.

На рисунку 3.45 зображено апаратне забезпечення цифрового годинника - термометра спроектоване в системі автоматизованого проектування та моделювання електронних схем та програмованих пристроїв Proteus VSM. На схемі 4 світлодіодних модуля 8x8 підключено до мікросхем-драйверів управління MAX7219 U1...U4. Мікросхема-драйвер U1 MAX7219 підключена до плати Arduino по інтерфейсу SPI. Вивід даних DIN мікросхеми підключено до виводу 11 (~PB3/**MOSI**/OC2A) DATA, вивід защіпки LOAD до виводу 10 (~PB2/$\overline{\text{SS}}$**SS**/OC1B), а вивід синхронізації CLK до виводу 13 (PB5/**SCK**) плати Arduino Uno/Micro. Виводи DP, A, B, C, D, E, F, G мікросхеми-драйвера U1 з'єднано відповідно з виводами XA0, XA1, XA2, XA3, XA4, XA5, XA6, XA7, а виводи DIG0, DIG1, DIG2, DIG3, DIG4, DIG5, DIG6, DIG7 з виводами YA0, YA1, YA2, YA3, YA4, YA5, YA6, YA7 першого матричного світлодіодного модуля 8x8. Аналогічно для мікросхем-драйверів U2…U4 виводи DP, A, B, C, D, E, F, G з'єднано з виводами відповідних світлодіодних модулів з XB0…XB7 другого світлодіодного модуля для U2, з виводами XC0…XC7 третього світлодіодного модуля для U3 і з виводами XD0…XD7 четвертого світлодіодного модуля для U4, а виводи DIG0, DIG1, DIG2, DIG3, DIG4, DIG5, DIG6, DIG7 з'єднано з YB0...YB7 для U2, YC0…YC7 для U3 і YD0…YD7 для U4. Вивід ISET мікросхем-драйверів підтягнуто через резистор 10 кОм до лінії живлення +5V. Вивід DOUT (вихід даних) кожної мікросхеми-драйвера з'єднано з виводом (входом даних) DIN наступної мікросхеми-драйвера (DOUT/DOUT1 першої мікросхеми-драйвера U1 з'єднано з входом даних DIN другої мікросхеми-драйвера U2, вихід DOUT2 другої мікросхеми-драйвера U2 з входом даних DIN третьої мікросхеми-драйвера U3 і т.д. каскадування

модулів). Таке підключення дозволяє нарощувати число матриць або модулів (каскадувати їх).

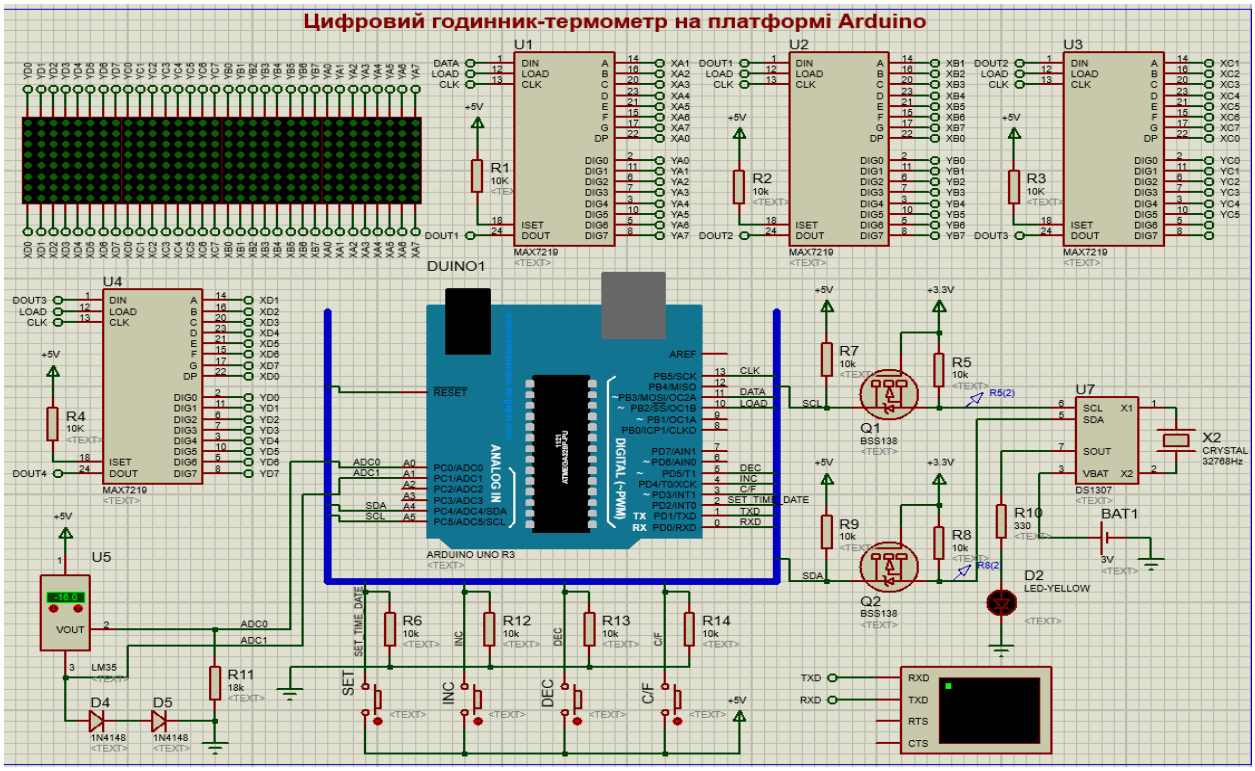


Рис.3.45. Апаратне забезпечення цифрового годинника-термометра спроектоване в САПР Proteus VSM

Аналоговий давач температури LM35DZ підключено за схемою вимірювання до аналогових входів ADC0 (PC0/ADC0) і ADC1 (PC1/ADC1) вбудованого в МК ATmega32u4 10-розрядного АЦП. Мікросхема годинника реального часу RTC DS1307 підключена до Arduino Micro по інтерфейсу I^2C. Лініями синхронізації (SCL) і даних (SDA) I^2C-інтерфейсу плати Arduino Micro є виводи A4 (PC4/ADC4/SDA) і A5 (PC5/ADC5/SCL). Кнопку встановити час і дату SET_TIME_DATE підключено до виводу 2 (PD2/INT0), кнопку Цельсія/Фаренгейт C_F до виводу 3 (~PD3/INT1), кнопку збільшити INC до виводу 4 (PD4/T0/XCK), кнопку зменшити DEC до виводу 5 (~PD5/T1).

РОЗДІЛ IV
ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЦИФРОВОГО ГОДИННИКА-ТЕРМОМЕТРА

4.1. Алгоритм роботи цифрового годинника-термометра

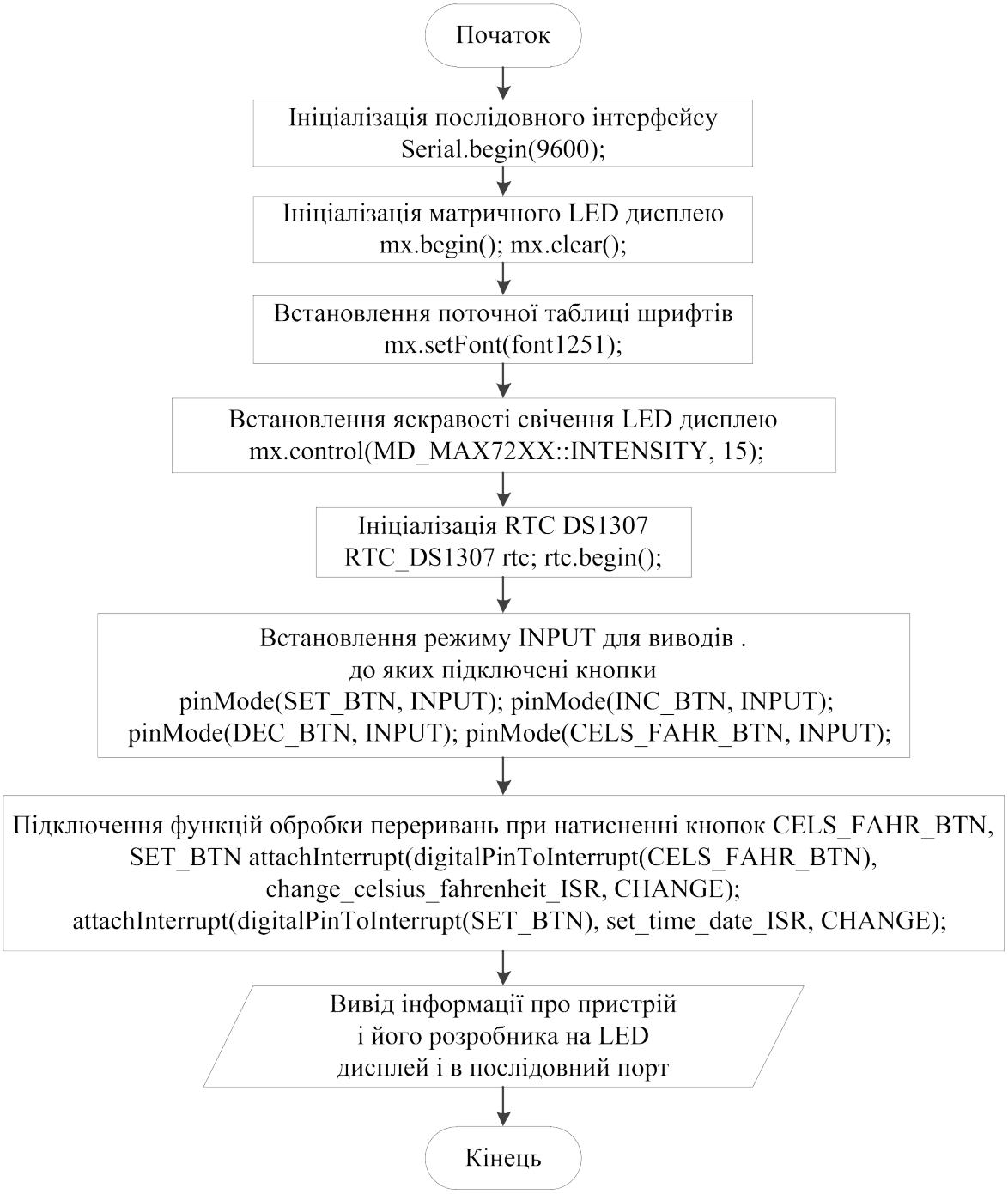
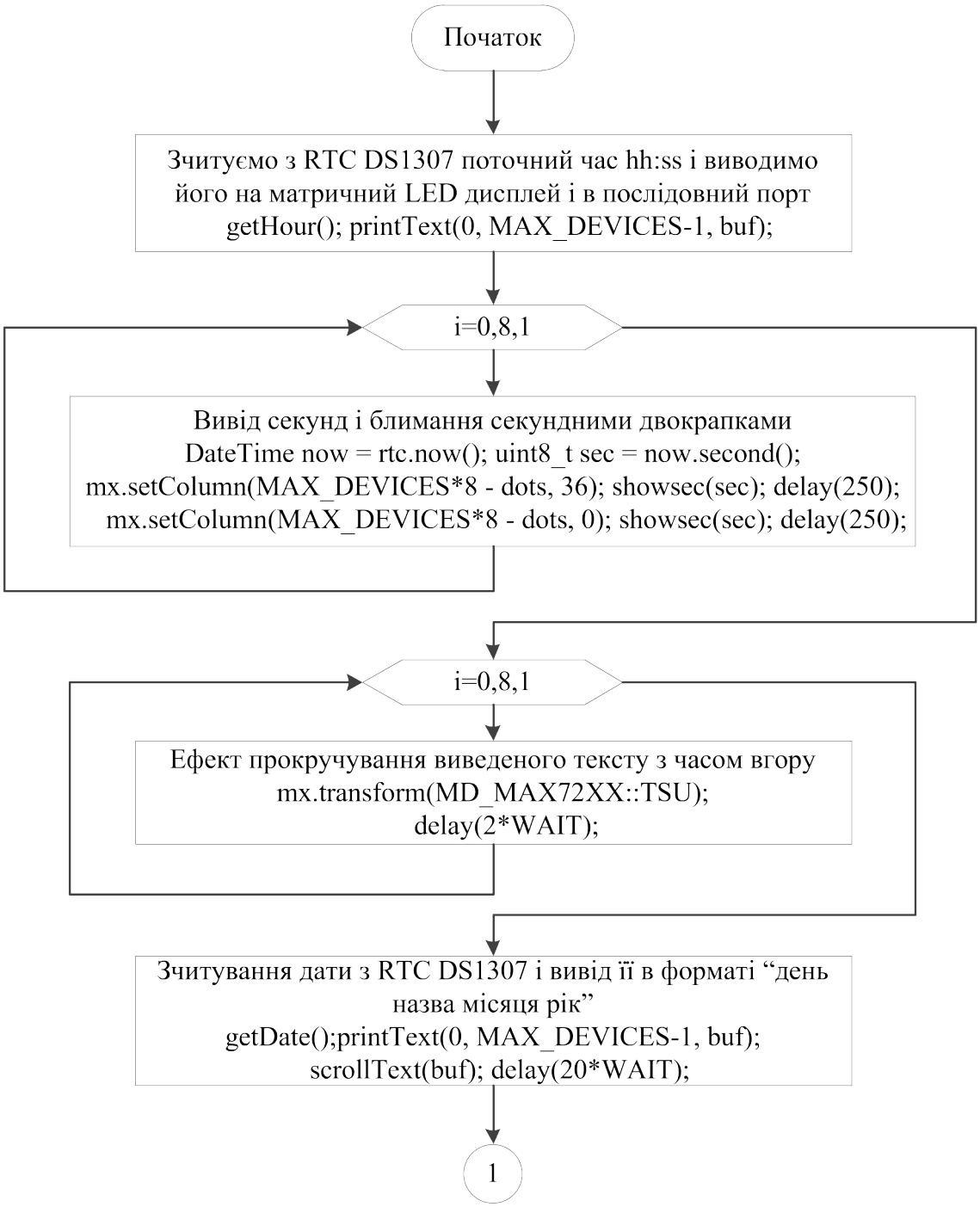


Рис.4.1. Алгоритм функції ініціалізації setup() цифрового годинника – термометра

На рисунку 4.1 зображено блок-схему алгоритму ініціалізації цифрового пристрою з функціями годинника і термометра.

На рисунку 4.2 зображено блок-схему алгоритму роботи цифрового годинника - термометра.



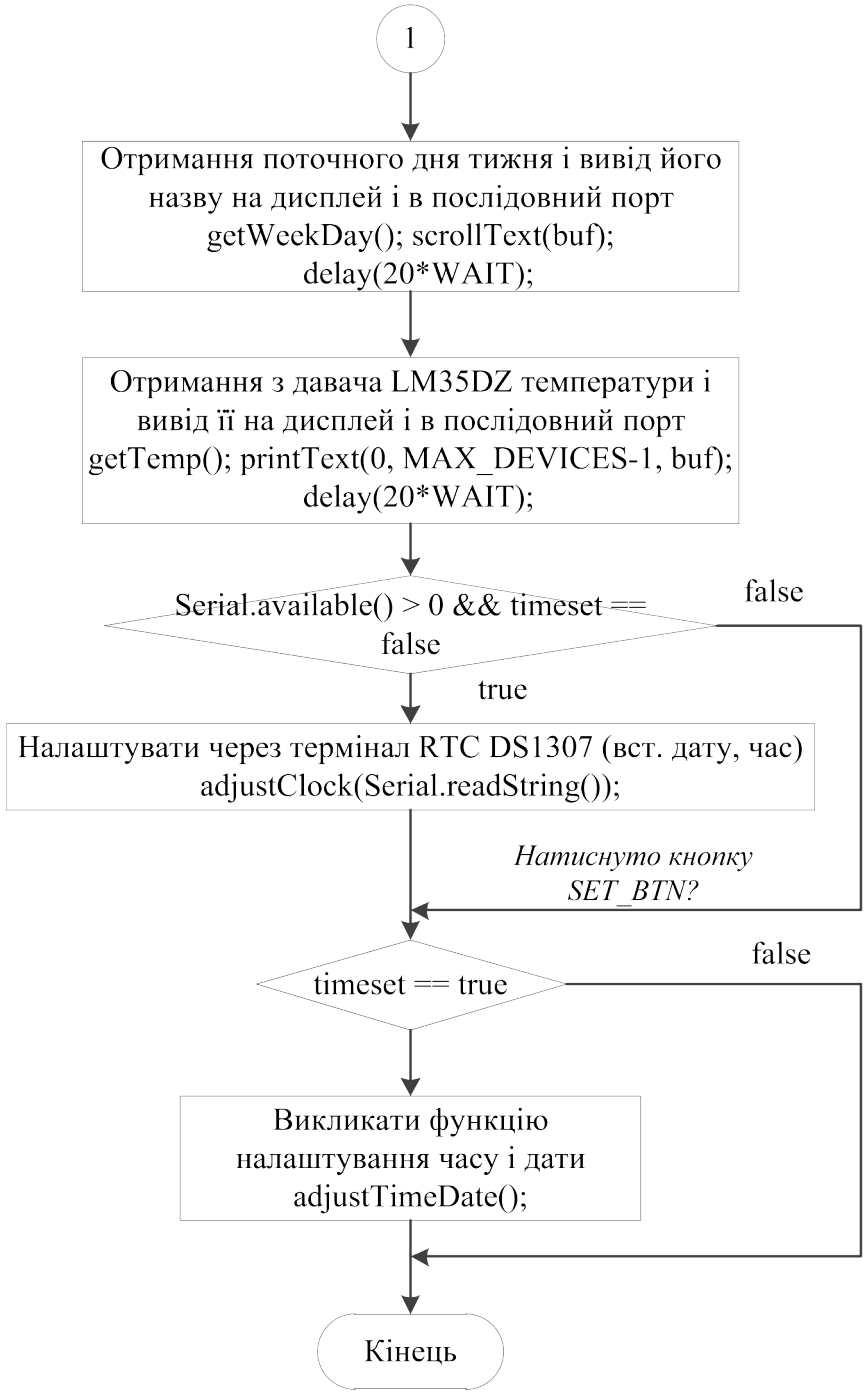


Рис.4.2. Алгоритм роботи цифрового годинника - термометра

На сьогоднішній час є декілька популярних бібліотек для управління світлодіодними матричними дисплеями MAX7219 з допомогою Arduino: MD_MAX72XX, MD_Parola, LedControl, MAXmatrix і т.д. Будемо використовувати бібліотеку MD_MAX72XX і бібліотеку MD_Parola для управління дисплеєм MAX7219. Для цього потрібно встановити ці бібліотеки в наше середовище розробки Arduino IDE. Щоб встановити ці бібліотеки,

потрібно перейти в Tools -> Manage Libraries… і знайти MD_MAX72XX і MD_Parola від MajicDesigns.

При увімкненні пристрою стартує записана у флеш-пам’ять мікроконтролера ATmega32u4 програма. Спочатку програма викликає функцію setup(), яка ініціалізує внутрішні змінні та периферію пристрою.

```
void setup(void)
{
```

У функції setup() створюється рядок title з заголовком про пристрій

```
 //char title[] = "Digital clock-thermometer based on Arduino platform ";
 //char author[] = "Developed by Denys Pyrozhenko";
 char title[] = "Цифровий годинник-термометр на платформі Arduino";
```

Створюється рядок author з текстом про розробника пристрою:

```
 char author[] = "Розробив: Денис Пироженко";
```

Проходить ініціалізація послідовного інтерфейсу з встановленням шикдкості обміну даними 9600 бод:

```
 Serial.begin(9600);
```

Вивід в послідовний порт текстових рядків title і author:

```
 Serial.println(F("Digital clock-thermometer"));
 Serial.println(F("based on Arduino, LM35 temperature sensor"));
 Serial.println(F("and 32x8 MAX7219 dot matrix LED display"));
 Serial.println(F("Developed by Denys Pyrozhenko"));
 Serial.println();
```

Вивід рядка в послідовний порт тексту про формат налаштування дати і часу через термінал послідовного порта:

```
 Serial.println(F(">> Use <dd/mm/yyyy hh:mm:ss> format to set clock's date and hour!"));
```

Ініціалізація матричного світлодіодного дисплйного модуля FC-16:

```
 mx.begin();
 mx.clear();
 mx.setFont(font1251);
 FONT_WIDTH = 5 + CHAR_SPACING; // The font width is 5 pixels
 mx.control(MD_MAX72XX::INTENSITY, 15);
```

Ініціалізація годинника реального часу DS1307:

```
 if (! rtc.begin()) {
```

```
  Serial.println("Couldn't find RTC");
  Serial.flush();
  abort();
 }

 if (! rtc.isrunning()) {
  Serial.println("RTC is NOT running, let's set the time!");
  // When time needs to be set on a new device, or after a power loss, the
  // following line sets the RTC to the date & time this sketch was compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
 }
 // When time needs to be re-set on a previously configured device, the
 // following line sets the RTC to the date & time this sketch was compiled
 // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
 // This line sets the RTC with an explicit date & time
 // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  setSQW(0x10); // 1 Гц
```

Ініціалізація пінів до яких підключені кнопки в режим входу (INPUT):

```
 pinMode(SET_BTN, INPUT);
 pinMode(INC_BTN, INPUT);
 pinMode(DEC_BTN, INPUT);
 pinMode(CELS_FAHR_BTN, INPUT);
```

Підключення функцій обробки переривання при натисненні кнопок CELS_FAHR_BTN і SET_BTN:

```
attachInterrupt(digitalPinToInterrupt(CELS_FAHR_BTN), change_celsius_fahrenheit_ISR, CHANGE);
attachInterrupt(digitalPinToInterrupt(SET_BTN), set_time_date_ISR, CHANGE);
```

Вивід рядка title з текстом про пристрій на матричний світлодіодний дисплей 4x8x8 з прокручуванням його справа наліво:

```
 scrollText(utf8cp866(title, buf));
 delay(WAIT); // пауза на WAIT секунд
```

Вивід рядка author з текстом про автора на матричний світлодіодний дисплей 4x8x8 з прокручуванням його справа наліво:

```
 scrollText(utf8cp866(author, buf));
}
```

Далі запускається головна функція loop() з нескінченним циклом виконання команд:

```
void loop() {
 mx.control(MD_MAX72XX::INTENSITY, 15);//ledIntensity);; // Use a value between 0 and 15 for brightness
```

Зчитування і вивід час hh:mm через buf на дисплей

```
   //delay(WAIT);
 // блимання двох крапок і виводимо секунди
 for (uint8_t i = 0; i < 8; i++){
  getHour();
  printText(0, MAX_DEVICES-1, buf);
  DateTime now = rtc.now();
  uint8_t sec = now.second();
  mx.setColumn(MAX_DEVICES*8 - dots, 36); // 00100100
  showsec(sec);
  delay(250);
  mx.setColumn(MAX_DEVICES*8 - dots, 0);
  showsec(sec);
  delay(250);
 }
 // вихід прокручуванням зображення догори
 for (uint8_t i = 0; i < 8; i++){
  mx.transform(MD_MAX72XX::TSU);
  delay(2*WAIT);
 }
 // зчитуємо і виводимо дату hh:mm через buf
 getDate();
 printText(0, MAX_DEVICES-1, buf);
 scrollText(buf);
 delay(20*WAIT);
```

```
  getWeekDay();
  scrollText(buf);
  delay(20*WAIT);
  getTemp();
  printText(0, MAX_DEVICES-1, buf);
  delay(20*WAIT);
```

Якщо є доступними дані через послідовний порт прапорець встановлення часу timeset є false, то викликається функція adjustClock для встанавлення нової дати і час, які передаються як рядковий параметр прочитаний з послідовного порта з допомогою функції Serial.readString():

```
  if (Serial.available() > 0 && timeset == false) {
    adjustClock(Serial.readString());
  }
  if (timeset == true)
  {
     detachInterrupt(digitalPinToInterrupt(SET_BTN));
     mx.clear();
     adjustTimeDate();
     mx.clear();
     timeset = false;
     attachInterrupt(digitalPinToInterrupt(SET_BTN), set_time_date_ISR, CHANGE);
  }
}
```

4.2. Розробка програмного модуля для роботи з давачем температури LM35DZ

Для отримання та виводу даних температури з давача температури LM35DZ створено функцію void getTemp(). Щоб визначити температуру спочатку потрібно виміряти напруги на виводах 2 і 3 давача і перетворити їх в цифровий код з допомогою вбудованого в мікроконтролер АЦП. Вивід давача 2 підключено до каналу A0/ADC0 АЦП, а вивід 3 давача підключено до каналу A1/ADC АЦП. В Arduino для читання значень (цифрових кодів) АЦП для виміряних напруг з давача температури здійснюється з використанням

програмної функції analogRead(pin), де pin – номер порта аналогового входу з якого буде здійснюватися зчитування (A0..A5 для більшості плат Arduino, 0..7 для Mini і 0..15 для Mega):

```
int adcVal1 = analogRead(LM35_V1); //ADC Read
int adcVal2 = analogRead(LM35_V2); //ADC Read
```

Далі прочитавши значення (цифрові коди) АЦП, знаючи його опорну напругу та роздільну здатність обчислюємо еквівалентні значення напруг:

```
float adcVolt1 = adcVal1 * (ADC_VREF_V / ADC_RESOLUTION);
float adcVolt2 = adcVal2 * (ADC_VREF_V / ADC_RESOLUTION); // ADC_VREF_mV = 2.56V, ADC_RESOLUTION = 1024 for all volt measurement. R1= 56K, R2=5K;
```

Далі за формулою знаючи значення напруг на виводах давача і віднявши значення adcVolt1 від adcVolt2 і помноживши на 100 отримаємо значення температури в градусах Цельсія:

```
float temp[2];
temp[0] = (adcVolt1 - adcVolt2)*100; // tempC
```

Перетворюємо значення температури з Цельсія у Фаренгейти

```
temp[1] = temp[0] * 9/5 + 32; // tempF
Serial.print(F("Temp: "));
Serial.print(temp[0]);   // вивести температуру в градусах Цельсія
Serial.print(F("\xB0\C, ")); // вивести символ градуса
Serial.print(temp[1]);   // вивести температуру в Фаренгейтах
Serial.println(F("\xB0\F")); // вивести знак градуса
 //char units[] = {'C', 'F'};
if ((temp[cf] > 0 && temp[cf] > 10) || (temp[cf] < 0 && temp[cf] < -10))
  sprintf(buf, "\x13%+d\x15", round(temp[cf]));//, units[cf]);
else if ((temp[cf] > 0 &&  temp[cf] < 10) || (temp[cf] < 0 && temp[cf] > -10))
  sprintf(buf, " \x13%+d\x15", round(temp[cf]));//, units[cf]);
if (!cf)
  strcat(buf, "C");
else
  strcat(buf, "F");
//sprintf(buf, " \x13%+d\x15\C", round(tempC));
//dtostrf(tempC, 3, 1, buf);
//strcat(buf, "\x13\C");
}
```

4.3. Розробка програмного модуля для роботи з годинником реального часу DS1307

Для роботи з мікросхемою годинника реального часу DS1307 розроблено бібліотеку RTClib, яка містить клас RTC_DS1307 з наступними методами:

bool begin (TwoWire *wireInstance=&Wire) – встановлює підключення по шині I^2C з DS1307 і повертає true, якщо зв'язок встановлено або false, якщо DS1307 не знайдено;

void adjust (const DateTime &dt) – встановлює задану дату і час в DS1307;

uint8_t isrunning (void) – перевіряє чи працює годинник DS1307? Перевіряє біт Clock Halt в регістрі 0;

DateTime now() – отримує поточну дату і час з DS1307. Повертає об'єкт класу DateTime;

static Ds1307SqwPinMode readSqwPinMode () – отримує значення поточного режиму роботи для піна SQW (CLKOUT);

void writeSqwPinMode (Ds1307SqwPinMode mode) – встановлює режим роботи для піна SQW (CLKOUT), який є виходом прямокутних імпульсів;

uint8_t readnvram (uint8_t address) – повертає байт, який зберігається в NVRAM за адресою address;

void readnvram (uint8_t *buf, uint8_t size, uint8_t address) - отримує значення з NVRAM в переданий буфер (масив) buf, починаючи з адреси address і наступні за ним адреси в кількості size-1;

void writenvram (uint8_t address, uint8_t data) – записує байт даних data в NVRAM за адресою address, address може приймати значення від 0 до 55, data – від 0 до 255;

void writenvram (uint8_t address, const uint8_t *buf, uint8_t size) – записує відразу декілька байтів в NVRAM. address – з якої адреси починається запис; size – кількість байтів для запису; buf – масив значень, які потрібно записати.

4.4. Розробка програмного модуля для роботи з матричним LED - дисплеєм FC-16

Для виводу інформації на матричний світлодіодний дисплей використано бібліотеку MD_MAX72xx, яка містить клас MD_MAX72xx. Нижче наведено опис використаних в основній програмі методів класу MD_MAX72xx:

Конструктор класу MD_MAX72xx MD_MAX72XX(moduleType_t mod, uint8_t csPin, uint8_t numDevices=1) за замовчуванням використовується інтерфейс SPI. Створює новий екземпляр класу (об'єкт класу). Можна створити декілька екземплярів класу, але вони не повинні використовувати один і той самий апаратний пін (вивід) CS (інтерфейс SPI). dataPin і clockPin визначаються апаратним розміщенням виводів SPI на платі Arduino. Параметр moduleType_t mod є одним зі значень enum moduleType_t, який визначає тип підключеного модуля. Ми mod задали через константу #define HARDWARE_TYPE MD_MAX72XX::FC16_HW. Параметр numDevices – кількість підключених пристроїв (матриць з драйверами MAX7219). Пам'ять для пристрою виділяється динамічно на базі параметра numDevices.

Метод класу void begin(void) ініціалізує дані об'єкта. Його потрібно викликати у функції setup() для ініціалізації нових даних класу, яку неможливо зробити при створенні об'єкта. Світлодіодний матричний модуль ініціалізується на середнє значення інтенсивності свічення світлодіодів, відображаються всі рядки, всі світлодіоди не світяться (виключені). Режими тестування, виключення і декодування відключені. Поновлення дисплею включено, а перенос відключено.

Метод inline void clear(void) { clear(0, getDeviceCount()-1); }очищує всі дисплейні дані на всіх дисплейних пристроях (матрицях).

Метод void clear(uint8_t startDev, uint8_t endDev) очищує всі дані дисплею на підмножині пристроїв (матриць). Параметр endDev має бути більшим або рівним startDev. Параметр startDev – перша матриця з IC

MAX7219, яка буде очищена [0..getDeviceCount()-1]. Параметр endDev остання матриця, яка буде очищена [0..getDeviceCount()-1].

Метод bool setColumn(uint16_t c, uint8_t value) { return setColumn((c / COL_SIZE), c % COL_SIZE, value) встановлює всі світлодіоди у визначеному ствопці у новий стан. Цей метод працює з одним стовпцем, встановлюючи значення світлодіодів в стовпці у вказане значення бітового поля. На стовпець зсилається абсолютний номер стовпця (тобто номер пристрою виводиться з стовпця). Цей метод зручний для виведення вертикальних ліній і узорів, коли дисплей розглядається як поле пікселів. Молодший значущий біт значення – це молодший номер рядка. Параметр uint16_t c є стопвпцем, який має бути встановленим [0..getColumnCount()-1]. Параметр uint8_t value встановлює значення кожного біта, якщо 1, то засвічується відповідний світлодіод. Повертає false при помилці параметрів, true в іншому випадку.

Метод inline bool setRow(uint8_t r, uint8_t value) { return setRow(0, getDeviceCount()-1, r, value); } встановлює всі світлодіоди підряд в новий стан на всіх пристроях (матрицях). Цей метод працює на всіх пристроях, встановлюючи значення світлодіодів в рядку у вказане значення бітового поля. Цей метод зручний для горизонтального відображення візерунків та ліній на всьому екрані. Молодший значущий біт значення – це молодший номер стовпця. Параметр r – рядок, який має бути встановленим [0..ROW_SIZE-1]. Параметр value містить значення кожного біта, якщо біт встановлено в 1, то буде засвічено відповідний світлодіод на кожному пристрої. Повертає false, якщо параметри помилкові, в іншому випадку true.

Метод bool setRow(uint8_t startDev, uint8_t endDev, uint8_t r, uint8_t value) встановлює всі світлодіоди підрял в новий стан на неперервній підножині пристроїв (матриць). Цей метод працює з неперервною підножиною пристроїв, встановлюючи значення світлодіодів в рядку у вказане значення бітового поля. Цей метод зручний для малювання візерунків та ліній по горизонталі тільки на визначених пристроях. Параметр endDev – має бути більшим або рівним startDev. Молодший значущий біт значення – це

молодший номер стовпця. Параметр startDev – це перший пристрій для перетворення [0..getDeviceCount()-1]. Параметр endDev – це останній пристрій для перетворення [0..getDeviceCount()-1]. Параметр r рядок, який має бути встановлений [0..ROW_SIZE-1]. Параметр value значення кожного біта, встановленого в 1, засвічує відповідний світлодіод на кожному пристрої. Повертає false, якщо параметри помилкові, в іншому випадку true.

Метод inline bool transform(transformType_t ttype) { return transform(0, getDeviceCount()-1, ttype); } використовується для перетворення даних на всіх пристроях. Буфери всіх пристроїв можна перетворити з допомогою одного з перерахованих перетворень в transformType_t. Перетворення здійснюється через границі пристроїв (при необхідності відбувається переповнення на сусідні пристрої (матриці)). Параметр ttype є одним з типів перетворення в transformType_t.

Метод bool transform(uint8_t startDev, uint8_t endDev, transformType_t ttype); використовується для перетворення даних в неперервній підножині пристроїв. Буфери для всіх пристроїв в підмножині можна перетворити з допомогою одного з перерахованих перетворень в transformType_t. Перетворення здійснюється через границі пристроїв (тобто при необхідності відбувається переповнення на сусідні пристрої). Параметр endDev має бути більщим або рівним startDev. Параметр startDev перший пристрій для перетворення [0..getDeviceCount()-1]. Параметр endDev останній пристрій для перетворення [0..getDeviceCount()-1]. Параметр ttype одне із типів перетворення в transformType_t. Функція повертає false, якщо параметри помилкові, в іншому випадку true.

enum MD_MAX72XX::transformType_t – це перерахований тип для визначення конкретного перетворення відображених даних в буфері пристрою. В таблиці 4.1 наведено способи перетворення дисплейних даних в буферах пристроїв.

Метод void update(controlValue_t mode) { control(UPDATE, mode); } – включає або виключає автоматичне оновлення дисплею. Коли автоматичне

оновлення відключено, то дисплей не буде оновлюватися після кожної операції. Оновлення дисплею можна примусуво викликати в будь-який час з допомогою виклику update() без параметрів. Ця функція представляє собою зручну оболонку для більш загального виклику функції control(). Параметр mode одне з значень типу controlValue_t (ON/OFF).

Таблиця 4.1

Перетворення дисплейних даних в буферах пристроїв

TSL	Перетворення зсуву на один елемент пікселя
TSR	Перетворення зсуву вправо на один елемент пікселя
TSU	Перетворення зсуву вгору на один елемент пікселя
TSD	Перетворення зсуву вниз на один елемент пікселя
TFLR	Перетворення відобразити зліва направо
TFUD	Перетворення відобразити згори вниз
TRC	Перетворення повернути на 90° за годинниковою стрілкою зліва направо
TINV	Перетворення інвертувати (пікселі інвертуються)

Метод void update(void) { flushBufferAll(); } примусово оновлює всі пристрої. Використовується, коли автоматичне оновлення відключено метоодом control. Це призведе до того, що всі буферизовані зміни будуть записані у всі підключені пристрої.

Метод bool clear(uint8_t buf) очищає всі відображувані дані у вказаному буфері. Параметр buf адреса буфера для очищення [0..getDeviceCount()-1]. Функція повертає false, якщо параметри помилкові, в іншому випадку true.

Метод uint8_t setChar(uint16_t col, uint16_t c) завантажує символ з даних шрифта, починаючи з визначеного стовпця. Завантажує символ з таблиці шрифтів безпосередньо на дисплей у вказаний стовпець. Поточна вибрана таблиця шрифтів використовується в якості джерела. Ця функція є доступною лише в тому випадку, якщо задане в бібліотеці значення USE_LOCAL_FONT встановлено в 1. Параметр col – номер стовпця в допустимому діапазоні [0..getDeviceCount()-1]. Параметр c – символ для виводу (відображення). Повертає ширину (в стовпцях) символа, 0 у випадку помилок параметра.

Метод bool control(uint8_t dev, controlRequest_t mode, int value) встановлює статус контролю (управління) вказаного параметра для вказаного пристрою. Пристрій має ряд параметрів контролю (управління), які можна встановити з допомогою цього методу. Тип управляючої дії потрібно передавати через параметр mode і має бути одним з дій управління (контролю) визначених controlRequest_t. Значення value, яке має бути вказане у потрібній дії управління, є одним з визначених дій в controlValue_t або числовим параметром, який підходить для дії управління. Параметр dev є адресою пристрою для управління [0..getDeviceCount()-1]. Параметр mode режим один з визначених запитів управління. Параметр value є значенням параметра або один з визначених статусів управління. Метод повертає false, якщо параметри помилкові, або true в іншому випадку.

Метод inline void control(controlRequest_t mode, int value) { control(0, getDeviceCount()-1, mode, value); } встановлює статус контролю вказаного параметра для всіх пристроїв. Викликає функцію управління для кожного пристрою по черзі, є обгорткою для методів управління (startDev, endDev, …). Параметр mode один з визначених запитів управління. Параметр value значення параметра або один із визначених статусів управління.

Метод bool control(uint8_t startDev, uint8_t endDev, controlRequest_t mode, int value) встановлює статус управління визначеного параметра для неперервної підмножини пристроїв. Викликає функцію управління для кожного пристрою по черзі для притсроїв в підножині. Параметр startDev – перший пристрій для управляючої дії [0..getDeviceCount()-1]. Параметр endDev – останній пристрій для управляючої дії [0..getDeviceCount()-1]. Параметр controlRequest_t mode один із визначених запитів управління. Повертає false, якщо параметри помилкові, в іншому випадку true.

4.5. Програмна реалізація алгоритму роботи цифрового годинника - термометра

На наступному лістингу наведено програмну реалізацію алгоритму роботи цифрового годинника – термометра:

```
void loop() {
  //byte ledIntensity = ledintensitySelect(analogRead(LDR_PIN));
  mx.control(MD_MAX72XX::INTENSITY, 15);//ledIntensity);; // Use a value between 0
and 15 for brightness
  // зчитуємо і виводимо час hh:mm через buf, блимання двох крапок і виводимо секунди
  for (uint8_t i = 0; i < 8; i++){
    getHour();
  }

  /*
getHour();
printText(0, MAX_DEVICES-1, buf);

  for (uint8_t i = 0; i < 8; i++){
    DateTime now = rtc.now();
    uint8_t sec = now.second();
    mx.setColumn(MAX_DEVICES*8 - dots, 36); // 00100100
    showsec(sec);
    delay(250);
    mx.setColumn(MAX_DEVICES*8 - dots, 0);
    showsec(sec);
    delay(250);
  }*/
  // вихід прокручуванням зображення догори
  for (uint8_t i = 0; i < 8; i++){
    mx.transform(MD_MAX72XX::TSU);
    delay(2*WAIT);
  }
  // зчитуємо і виводимо дату hh:mm через buf
  getDate();
  printText(0, MAX_DEVICES-1, buf);
  scrollText(buf);
  delay(20*WAIT);
  getWeekDay();
  scrollText(buf);
  delay(20*WAIT);
  getTemp();
  printText(0, MAX_DEVICES-1, buf);
  delay(20*WAIT);

  // Time setting in RTC:
  if (Serial.available() > 0 && timeset == false) {
    adjustClock(Serial.readString());
  }
  if (timeset == true)
  {
     detachInterrupt(digitalPinToInterrupt(SET_BTN));
     mx.clear();
     adjustTimeDate();
     mx.clear();
     timeset = false;
     attachInterrupt(digitalPinToInterrupt(SET_BTN), set_time_date_ISR, CHANGE);
  }
}
```

4.6. Моделювання та дослідження макету цифрового годинника – термометра

Скомпільована програма для МК сімейства AVR є файлом типу cof, який прошивається в МК. Спроектовану схему цифрового пристрою і роботу прошивки перевіряємо в симуляторі Proteus ISIS. ProteusVSM - це потужне ПЗ для моделюванняі налагодження досить складних пристроїв в яких може міститися кілька МК одночасно і навіть різних родин в одному пристрої. На рисунках нижче зображено моделювання роботи спроектованого цифрового пристрою. PROTEUS VSM - пакет програм для автоматизованого проектування (САПР) електронних схем. Розробка компанії Labcenter Electronics (Великобританія). PROTEUS VSM маєможливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP і інш. Пакет Proteus складається з двохчастин, двох підпрограм: ISIS- програма синтезу та моделювання безпосередньо електронних схем і ARES - програма розробки друкованих плат.

Моделювання цифрового годинника-термометра в Proteus ISIS подано в додатку 2.

Дослідження макету цифрового годинника-термометра подано в додатку 3.

ВИСНОВКИ

В дипломній роботі розроблено апаратне та програмне забезпечення цифрового годинника-термометра на платформі Arduino Micro і давачі температури LM35DZ. Цифровий пристрій зчитує час і дату з мікросхеми RTC (годинника реального часу) DS1307, температуру з давача LM35DZ та відображає їх на дисплейному модулі з 4 світлодіодних матриць, що управляються мікросхемами – драйверами MAX7219. Пристрій має можливість налаштувати час, дату та змінити шкалу температури з градусів Цельсія на градуси Фаренгейта і навпаки.

При розробці цифрового годинника - термометра використано сучасну елементну базу. Основними її елементами є плата Arduino Micro на МК AVR ATmega32u4 (Arduino Uno на МК ATmega328P), годинник реального часу (RTC) DS1307, прецизійний давач температури за Цельсієм LM35DZ, дисплейний модуль FC-16 з 4 точкових світлодіодних матриць і на базі мікросхем – драйверів MAX7219.

Спроектовано електричну принципову схему та створено модель цифрового годинника-термометра в Proteus VSM. Розроблено алгоритм роботи і програмне забезпечення цифрового пристрою на мовах C/C++ з використанням інструментарію розробки Arduino IDE. Розроблено програмні модулі для роботи з годинником реального часу RTC DS1307, давачем температури LM35DZ, та програмний модуль виводу інформації на матричний світлодіодний дисплейний модуль FC-16. Проведено моделювання роботи цифрового годинника - термометра в емуляторі Proteus ISIS. Результати моделювання показали коректність програмної реалізації алгоритму роботи цифрового пристрою та розроблених програмних модулів. Зібрано та досліджено макет цифрового годинника-термометра.

Під час написання дипломної роботи здобув нові знання і вдосконалив свої практичні навики з розробки цифрових програмованих пристроїв на мікроконтролерах, розробки алгоритмів, програмування та відладки програм для вбудованих систем.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Massimo Banzi, Michael Shiloh. Getting Started with Arduino, 3rd Edition. Published by Maker Media, Inc., 2015, p. 262.
2. Ryan Turner. Arduino Programming: The Ultimate Intermediate Guide to Learn Arduino Programming Step by Step. Publisher: nelly B.L. International Consulting LTD. (December 5, 2019), p. 164.
3. Joe Padrue. C Programming for microcontrollers. 2005. – 300 с.
4. Електронні ресурси з навчальними матеріалами по Arduino: https://www.arduino.cc/en/Tutorial/HomePage, https://www.tutorialspoint.com/arduino/index.htm, https://alexgyver.ru/arduino_lessons/, https://www.programmingelectronics.com/tutorial-3-arduino-ide-and-sketch-overview/, https://www.javatpoint.com/arduino.
5. Електронний ресурс з програмним забезпеченням Arduino IDE: https://www.arduino.cc/en/software.
6. Електронні ресурси з описом плати Arduino Micro: https://www.farnell.com/datasheets/1685581.pdf, https://www.etechnophiles-.com/arduino-micro-pinout-schematic-and-specifications/, https://docs.rs-online.com/0a47/0900766b8119187a.pdf, https://docs.arduino.cc/hardware-/micro.
7. Електронний ресурс на МК ATmega32u4: https://ww1.microchip.com/-downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf, http://ww1.microchip.com/downloads/en/devicedoc/-Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Summary.pdf.
8. Електронний ресурс datasheet на RTC DS1307: https://www.analog.com/-media/en/technical-documentation/data-sheets/DS1307.pdf.
9. Електронний ресурс datasheet на давач LM35: https://www.ti.com/lit/-ds/symlink/lm35.pdf, https://www.electroschematics.com/wp-content/uploads/-2010/02/LM35-DATASHEET.pdf.

10. Електронні ресурси по використанню давача LM35 з Arduino: https://www.makerguides.com/lm35-arduino-tutorial/, https://lastminuteengineers.com/lm35-temperature-sensor-arduino-tutorial/, https://arduinogetstarted.com/tutorials/arduino-lm35-temperature-sensor.
11. Електронний ресурс на datasheet по мікросхемі MAX7219: https://www.analog.com/media/en/technical-documentation/data-sheets/max7219-max7221.pdf, https://www.sparkfun.com/datasheets/-Components/General/COM-09622-MAX7219-MAX7221.pdf.
12. Електронні ресурси по використанню MAX7219 матричного світлодіодного дисплею з Arduino: https://www.circuitgeeks.com/arduino-max7219-led-matrix-display/, https://circuitdigest.com/microcontroller-projects/interfacing-max7219-led-dot-matrix-display-with-arduino, https://www.instructables.com/User-Manual-MAX7219-Dot-Matrix-4-in-1/, https://lastminuteengineers.com/max7219-dot-matrix-arduino-tutorial/, https://projecthub.arduino.cc/mdraber/0c417a04-ec3f-405a-a383-b2d66e889e7a, https://www.makerguides.com/max7219-led-dot-matrix-display-arduino-tutorial/, https://howtomechatronics.com/tutorials/arduino/8x8-led-matrix-max7219-tutorial-scrolling-text-android-control-via-bluetooth/, .

ДОДАТКИ

Додаток 1

Лістинг головного програмного модуля цифрового годинника-термометра

```
/*
 Цифровий годинник-термометр на платформі Arduino і давачі температури LM35DZ
 (Digital clock-thermometer based on Arduino platform and LM35DZ temperature sensor)
 Розробив: студент 4 курсу, Пироженко Денис
*/
#include <MD_MAX72xx.h>
#include <SPI.h>
#include "Parola_Fonts_data.h"
#include <Wire.h>
#include "RTClib.h"

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW //PAROLA_HW
#define MAX_DEVICES 4

#define CLK_PIN   13
#define DATA_PIN  11
#define CS_PIN    10

// апаратний інтерфейс підключення SPI
MD_MAX72XX mx = MD_MAX72XX(HARDWARE_TYPE, CS_PIN, MAX_DEVICES);

#define BUF_SIZE      200  // розмір текстового буфера
#define CHAR_SPACING  1    // кількість пікселів між символами
#define DELAYTIME 50
#define WAIT 100

char buf[BUF_SIZE], secs[4];
uint8_t dots;
byte FONT_WIDTH;

#define ADC_VREF_V    5.0 // у вольтах
#define ADC_RESOLUTION 1023.0
#define PIN_LM35       A0
#define LM35_V1 A0
#define LM35_V2 A1

// піни до яких підключені кнопки
#define NUM_OF_BTNS 4
#define SET_BTN 2
#define CELS_FAHR_BTN 3
#define INC_BTN 4
#define DEC_BTN 5

unsigned long lastDebounceTime[NUM_OF_BTNS] = {0, 0, 0, 0};

int lastBtnState[NUM_OF_BTNS] = {LOW, LOW, LOW, LOW};
int btnState[NUM_OF_BTNS];
int btnsArray[NUM_OF_BTNS] = {SET_BTN, INC_BTN, DEC_BTN};
unsigned long debounceDelay = 20;

RTC_DS1307 rtc;
volatile bool timeset = false;

//const char daysOfTheWeek[7][12] = {/*"Sunday"*/ "Неділя", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"};
//char *weekdays[]={"Sun","Mon","Tue","Wed","Thr","Fri","Sat"};
```

```
//char
*monthes[]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
volatile uint8_t cf = 0;

void change_celsius_fahrenheit_ISR() {
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (digitalRead(CELS_FAHR_BTN) == HIGH)
  {
    if (cf)
      cf = 0;
    else
      cf = 1;
  }
}

// button array pos
int btnToPos(uint8_t btn)
{
   for (int i = 0; i < NUM_OF_BTNS; i++) {
     if (btnsArray[i] == btn)
       return i;
   }
   return -1;
}

bool isBtnPressed(int btn_pin)
{
  // читаємо пін кнопки, якщо кнопка натиснута буде високим, якщо ні буде низьким
  int value = digitalRead(btn_pin);
  int pos = btnToPos(btn_pin);
  if (value != lastBtnState[pos])  {
    lastDebounceTime[pos] = millis();
  }
  // перевірити різницю між поточним часом і останнім зареєстрованим часом натиснення
кнопки, якшо він більший встановленого часу затримки, то означає що натиснута кнопка
  if ((millis() - lastDebounceTime[pos]) > debounceDelay) {
    if (value != btnState[pos]) {
      btnState[pos] = value;
      if (btnState[pos] == HIGH)
        return true;
    }
  }
  //зберегти зчитане значення стан кнопки. наступного ращу в циклі зчитаний стан
кнопки буде вже як lastButtonState
  lastBtnState[pos] = value;
  return false;
}

char* utf8cp866(char* src, char *dest)
{
  int i = 0, len = strlen(src);
  String target;
  unsigned char n;
  char tmp[2] = { '0', '\0' };
  strcpy(dest, "");
  while (i < len) {
    n = src[i]; i++;
    //if (n >= 0xC0) {
      switch (n) {
        case 0xD0: {
          n = src[i]; i++;
          if (n >= 0x90 && n <= 0xBF) n = n - 0x10;
```

```
          break;
        }
        case 0xD1: {
          n = src[i]; i++;
          //if (n == 0x91) { n = 0xB8; break; }
          if (n >= 0x80 && n <= 0x8F) n = n + 0x30;
          break;
        }
      }
    //}
    tmp[0] = n;
    strcat(dest, tmp);
  }
  return dest;
}

void adjustTimeDate() {
  uint8_t set_flag = false;
  DateTime now = rtc.now();
  uint8_t _hour, _minute, _second;
  uint8_t _day, _month;
  uint16_t _year;
  //scrollText(utf8cp866("Налаштування часу", buf));
  _hour = now.hour();
  char str[5];
  utf8cp866(" ГГ: ", str);
  sprintf(buf, "%s%02u", str, _hour);
  printText(0, MAX_DEVICES-1, buf);
  delay(100);
  // hours
  while (!set_flag)
  {
    if (isBtnPressed(INC_BTN))
    {
       if (_hour < 23)
         _hour++;
       else _hour = 0;
       sprintf(buf, "%s%02u", str, _hour);
       printText(0, MAX_DEVICES-1, buf);
    }
    if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
    {
      if (_hour > 0)
        _hour--;
      else _hour = 23;
      sprintf(buf, "%s%02u", str, _hour);
      printText(0, MAX_DEVICES-1, buf);
    }
    if (isBtnPressed(SET_BTN))
      set_flag = true;
  }
  // minutes
  _minute = now.minute();
  set_flag = false;
  //sprintf(buf, "%02u", _minute);
  utf8cp866(" XX: ", str);
  sprintf(buf, "%s%02u", str, _minute);
  printText(0, MAX_DEVICES-1, buf);
  while (!set_flag)
  {
    if (isBtnPressed(INC_BTN))
    {
```

```
      if (_minute < 59)
        _minute++;
      else _minute = 0;
      sprintf(buf, "%s%02u", str, _minute);
      printText(0, MAX_DEVICES-1, buf);
    }
    if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
    {
      if (_minute > 0)
        _minute--;
      else _minute = 59;
      sprintf(buf, "%s%02u", str, _minute);
      printText(0, MAX_DEVICES-1, buf);
    }
    if( isBtnPressed(SET_BTN))
      set_flag = true;
}
// seconds
_second = now.second();
set_flag = false;
//sprintf(buf, "%02u", _second);
utf8cp866(" СС: ", str);
sprintf(buf, "%s%02u", str, _second);
printText(0, MAX_DEVICES-1, buf);
while (!set_flag)
{
  if (isBtnPressed(INC_BTN))
  {
    if (_second < 59)
      _second++;
    else _second = 0;
    sprintf(buf, "%s%02u", str, _second);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
  {
    if (_second > 0)
      _second--;
    else _second = 59;
    sprintf(buf, "%s%02u", str, _second);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(SET_BTN))
   set_flag = true;
}
// day
_day = now.day();
set_flag = false;
utf8cp866(" ДД: ", str);
//sprintf(buf, "%02u", _day);
sprintf(buf, "%s%02u", str, _day);
printText(0, MAX_DEVICES-1, buf);
while (!set_flag)
{
  if (isBtnPressed(INC_BTN))
  {
    if (_day < 30)
      _day++;
    else _day = 1;
    sprintf(buf, "%s%02u", str, _day);
    printText(0, MAX_DEVICES-1, buf);
  }
```

```
  if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
  {
    if (_day > 1)
      _day--;
    else _day = 31;
    sprintf(buf, "%s%02u", str, _day);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(SET_BTN))
   set_flag = true;
}
// month
_month = now.month();
set_flag = false;
utf8cp866(" MM: ", str);
sprintf(buf, "%s%02u", str, _month);
printText(0, MAX_DEVICES-1, buf);
while (!set_flag)
{
  if (isBtnPressed(INC_BTN))
  {
    if (_month < 11)
      _month++;
    else _month = 1;
    sprintf(buf, "%s%02u", str, _month);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
  {
    if (_month > 1)
      _month--;
    else
      _month = 12;
    sprintf(buf, "%s%02u", str, _month);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(SET_BTN))
   set_flag = true;
}
// year
_year = now.year();
uint16_t yy = _year / 100;
yy *= 100;
_year %= 100;
Serial.println(yy);
Serial.println(_year);
set_flag = false;
utf8cp866(" PP: ", str);
sprintf(buf, "%s%02u", str, _year);
printText(0, MAX_DEVICES-1, buf);
while (!set_flag)
{
  if (isBtnPressed(INC_BTN))
  {
    if (_year < 99)
      _year++;
    else
      _year = 0;
    sprintf(buf, "%s%02u", str, _year);
    printText(0, MAX_DEVICES-1, buf);
  }
  if (isBtnPressed(DEC_BTN)) // натиснута кнопка SELECT_MINUS
```

```
    {
      if (_year > 0)
        _year--;
      else
        _year = 99;
      sprintf(buf, "%s%02u", str, _year);
      printText(0, MAX_DEVICES-1, buf);
    }
    if (isBtnPressed(SET_BTN))
     set_flag = true;
  }
  Serial.println(_year+yy);
  rtc.adjust(DateTime(_year+yy, _month, _day, _hour, _month, _second));
}

void set_time_date_ISR() {
  /*buttonState = digitalRead(buttonPin);
  digitalWrite(ledPin, buttonState);.*/
  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (digitalRead(SET_BTN) == HIGH) //isBtnPressed(MENU_BTN_PIN))
  {
     timeset = true;
  }
}

void adjustClock(String data) {
  uint8_t _day = data.substring(0,2).toInt();
  uint8_t _month = data.substring(3,5).toInt();
  uint16_t _year = data.substring(6,10).toInt();
  uint8_t _hour = data.substring(11,13).toInt();
  uint8_t _minute = data.substring(14,16).toInt();
  uint8_t _second = data.substring(17,19).toInt();
  rtc.adjust(DateTime(_year, _month, _day, _hour, _minute, _second));
  Serial.println(F(">> Datetime successfully set!"));
  timeset = true;
}

void printText(uint8_t modStart, uint8_t modEnd, uint8_t *pMsg)
{
  uint8_t   state = 0;
  uint8_t   curLen;
  uint16_t  showLen;
  uint8_t   cBuf[FONT_WIDTH];
  int16_t   col = ((modEnd + 1) * COL_SIZE) - 1;
  mx.control(modStart, modEnd, MD_MAX72XX::UPDATE, MD_MAX72XX::OFF);
  do     // finite state machine to print the characters in the space available
  {
    switch(state)
    {
      case 0: // Load the next character from the font table
        // if we reached end of message, reset the message pointer
        if (*pMsg == '\0')
        {
          showLen = col - (modEnd * COL_SIZE);  // padding characters
          state = 2;
          break;
        }
        // retrieve the next character form the font file
        showLen = mx.getChar(*pMsg++, sizeof(cBuf)/sizeof(cBuf[0]), cBuf);
        curLen = 0;
        state++;
      case 1: // display the next part of the character
```

```
        mx.setColumn(col--, cBuf[curLen++]);
        // done with font character, now display the space between chars
        if (curLen == showLen)
        {
          showLen = CHAR_SPACING;
          state = 2;
        }
        break;
      case 2: // initialize state for displaying empty columns
        curLen = 0;
        state++;
      case 3:  // display inter-character spacing or end of message padding (blank
columns)
        mx.setColumn(col--, 0);
        curLen++;
        if (curLen == showLen)
          state = 0;
        break;
      default:
        col = -1;   // this definitely ends the do loop
    }
  } while (col >= (modStart * COL_SIZE));
  mx.control(modStart, modEnd, MD_MAX72XX::UPDATE, MD_MAX72XX::ON);
}

void scrollText(char *p)   // copied from library
{
  uint8_t charWidth;
  uint8_t cBuf[8];  // this should be ok for all built-in fonts
  //char *s = p;
  mx.clear();

  while (*p != '\0')
  {
    charWidth = mx.getChar((uint8_t)*p++, sizeof(cBuf) / sizeof(cBuf[0]), cBuf);
    for (uint8_t i = 0; i <= charWidth; i++) // allow space between characters
    {
      mx.transform(MD_MAX72XX::TSL);
      if (i < charWidth)
      {
        uint16_t c = cBuf[i];
        mx.setColumn(0, c);//Buf[i]);
      }
      delay(DELAYTIME);
    }
  }
  for (uint8_t i = 0; i < 8*MAX_DEVICES; i++){
    mx.transform(MD_MAX72XX::TSL); delay(DELAYTIME);
  }
}

void setSQW(uint8_t value) {
  Wire.beginTransmission(0x68);
  Wire.write(7);
  Wire.write(value);
  Wire.endTransmission();
}

void setup(void)
{
  //char title[] = "Digital clock-thermometer based on Arduino platform";
  //char author[] = "Developed by Denys Pyrozhenko";
```

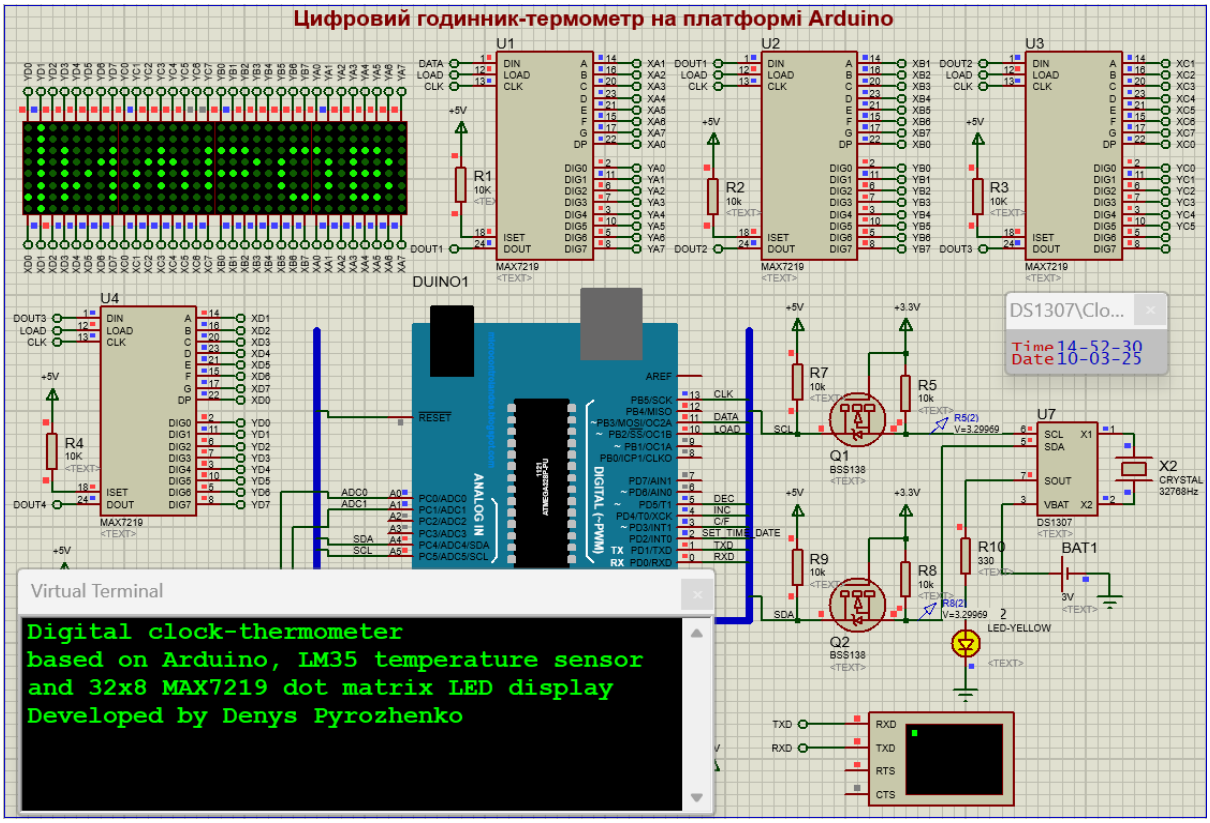
```
  char title[] = "Цифровий годинник-термометр на платформі Arduino";
  char author[] = "Розробив: Пироженко Денис";
  Serial.begin(9600);
  /*Serial.println(F("Digital clock-thermometer"));
  Serial.println(F("based on Arduino, LM35 temperature sensor"));
  Serial.println(F("and 32x8 MAX7219 dot matrix LED display"));
  Serial.println(F("Developed by Denys Pyrozhenko"));*/
  Serial.println(F("Цифровий годинник-термометр на платформі Arduino, давачі LM35"));
  //Serial.println(F("на платформі Arduino, давачі LM35"));
  Serial.println(F("та 32x8 MAX7219 матричному світлодіодному дисплеї"));
  Serial.println(F("Розробив: Денис Пироженко"));
  Serial.println();
  Serial.println(F(">> Для встановлення дати і часу використовується формат
<dd/mm/yyyy hh:mm:ss> !"));
  mx.begin();
  mx.clear();
  mx.setFont(font1251);
  FONT_WIDTH = 5 + CHAR_SPACING; // The font width is 5 pixels
  mx.control(MD_MAX72XX::INTENSITY, 15);
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }
  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
   setSQW(0x10); // 1 Гц
  // ініціалізація пінів кнопок як входи:
  pinMode(SET_BTN, INPUT);
  pinMode(INC_BTN, INPUT);
  pinMode(DEC_BTN, INPUT);
  pinMode(CELS_FAHR_BTN, INPUT);
  // підключаємо функцію обробки переривання - Attach an interrupt to the ISR vector
  attachInterrupt(digitalPinToInterrupt(CELS_FAHR_BTN),
change_celsius_fahrenheit_ISR, CHANGE);
  attachInterrupt(digitalPinToInterrupt(SET_BTN), set_time_date_ISR, CHANGE);
  scrollText(utf8cp866(title, buf));
  delay(WAIT);
  scrollText(utf8cp866(author, buf));
}

void getDate() // зчитування дати з RTC DS1307
{
  //char* months[] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"};
  //char* months[] = {"Січ.", "Лют.", "Берез.", "Квіт.", "Трав.", "Черв.", "Лип.",
"Серп.", "Верес.", "Жовт.", "Листоп.", "Груд."};
  //char* months[] = {"Січня", "Лютого", "Березня", "Квітня", "Травня", "Червня",
"Липня", "Серпня", "Вересня", "Жовтня", "Листопада", "Грудня"};
  char* months[] = {"січня", "лютого", "березня", "квітня", "травня", "червня",
"липня", "серпня", "вересня", "жовтня", "листопада", "грудня"};
  char tmp[9];
  DateTime now = rtc.now();
  uint8_t _day = now.day(); // отримання дня
  uint8_t _month = now.month(); // отримання місяця
  uint16_t _year = now.year();
  sprintf(buf, "%02u.%02u.%u%u", _day, _month, (_year / 10) % 10, _year % 10);
  Serial.print(F("Дата: "));
```

```
  Serial.println(buf);
  sprintf(buf, "%u %s %u ", _day, utf8cp866(months[_month - 1], tmp), _year);//
(_year / 10) % 10, _year % 10);
  strcat(buf, utf8cp866("року", tmp));
}

void getWeekDay()
{
  char* weekdays[] = {"Неділя", "Понеділок", "Вівторок", "Середа", "Четвер",
"П'ятниця", "Субота"};
  char tmp[9];
  DateTime now = rtc.now();
  uint8_t wd = now.dayOfTheWeek();
  sprintf(buf, "%s", utf8cp866(weekdays[wd], tmp));
}

void getHour()
{
  // зчитування годин і хвилин з RTC DS1307
  DateTime now = rtc.now();
  uint8_t _hour = now.hour();
  uint8_t _minute = now.minute();
  uint8_t _second = now.second();
  if (_hour < 10) dots = 7;
  if(_hour > 19 && _hour < 24)
    dots = 13;
  if ((_hour > 9 && _hour < 20) || (_hour == 21))
    dots = 11;
  if (_hour == 1) dots = 5;
  if (_hour == 11) dots = 10;
  sprintf(buf, "%02u:%02u", _hour, _minute);
  printText(0, MAX_DEVICES-1, buf);
  //Serial.print(F("Time: "));
  Serial.print(F("Час: "));
  Serial.print(buf);
  Serial.print(F(":"));
  Serial.print(_second / 10);
  Serial.println(_second % 10);
  // блимання двох крапок і виводимо секунди
  mx.setColumn(MAX_DEVICES*8 - dots, 36); // 00100100
  showsec(_second);
  delay(250);
  mx.setColumn(MAX_DEVICES*8 - dots, 0);
  showsec(_second);
  delay(250);
}

void showsec(uint8_t sec)
{
  uint8_t sec1 = 192 + sec/10; // перетворення перщої цифри секунд в символьне
представлення для виводу на матричний LED дисплей
  uint8_t sec2 = 192 + sec%10; // перетворення другої цифри секунд в символьне
представлення для виводу на матричний LED дисплей
  mx.setChar(6, sec1);
  mx.setChar(2, sec2);
}

void getTemp()
{
  // отримання значення ADC з давача температури
  int adcVal1 = analogRead(LM35_V1); //ADC Read
  int adcVal2 = analogRead(LM35_V2); //ADC Read
```

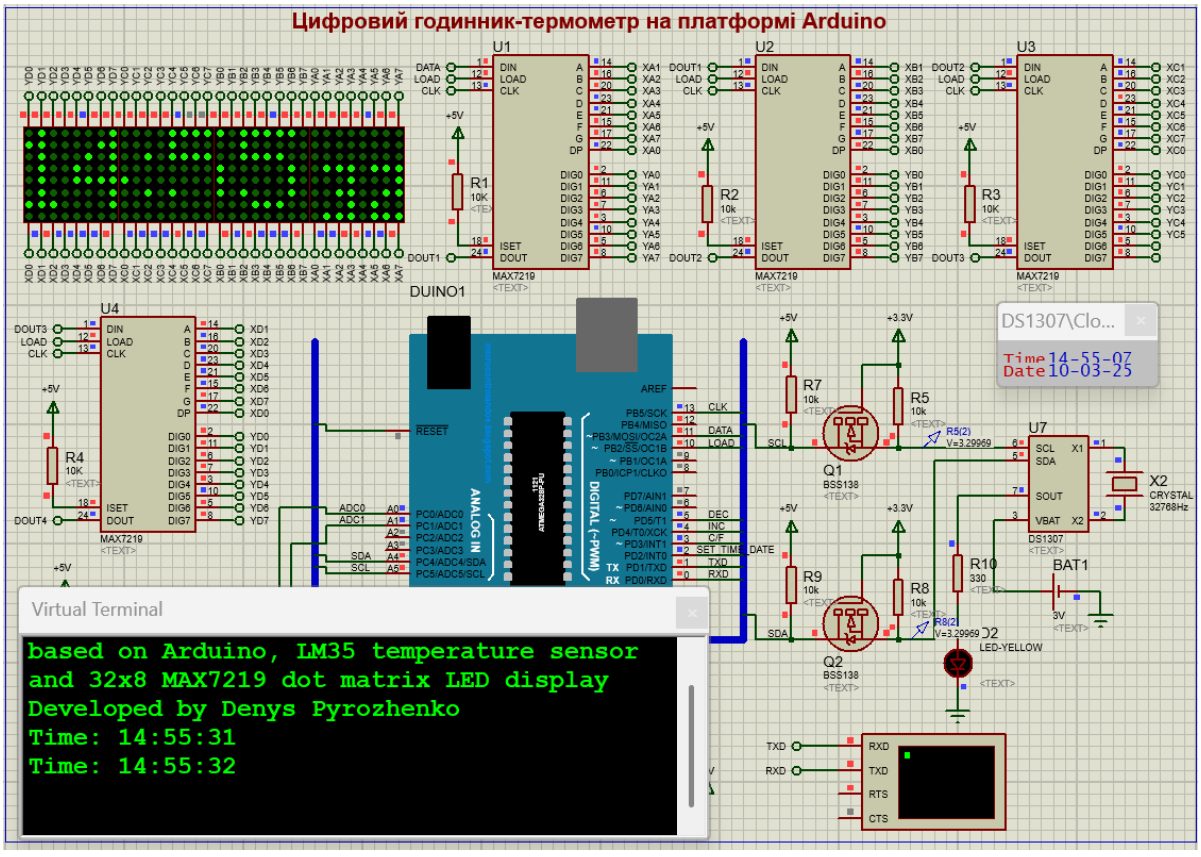
```
  float adcVolt1 = adcVal1 * (ADC_VREF_V / ADC_RESOLUTION); // ADC_VREF_mV = 2.56V,
ADC_RESOLUTION = 1024 for all volt measurement. R1= 56K, R2=5K;
  float adcVolt2 = adcVal2 * (ADC_VREF_V / ADC_RESOLUTION); // ADC_VREF_mV = 2.56V,
ADC_RESOLUTION = 1024 for all volt measurement. R1= 56K, R2=5K;
  float temp[2];
  temp[0] = (adcVolt1 - adcVolt2)*100; // tempC
  // перетворення значення температури з Цельсія у Фаренгейти
  temp[1] = temp[0] * 9/5 + 32; // tempF
  //Serial.print(F("Temp: "));
  Serial.print(F("Температура: "));
  Serial.print(temp[0]);   // print the temperature in Celsius
  Serial.print(F("\xC2\xB0\C, ")); // shows degree symbol \xB0
  Serial.print(temp[1]);   // print the temperature in Fahrenheit
  Serial.println(F("\xC2\xB0\F")); // shows degree symbol
   //char units[] = {'C', 'F'};
  if ((temp[cf] > 0 && temp[cf] > 10) || (temp[cf] < 0 && temp[cf] < -10))
    sprintf(buf, "\x13%+d\x15", round(temp[cf]));//, units[cf]);
  else if ((temp[cf] > 0 &&  temp[cf] < 10) || (temp[cf] < 0 && temp[cf] > -10))
    sprintf(buf, " \x13%+d\x15", round(temp[cf]));//, units[cf]);
  if (!cf)
    strcat(buf, "C");
  else
    strcat(buf, "F");
}

void loop() {
  mx.control(MD_MAX72XX::INTENSITY, 15 // Use a value between 0 and 15 for brightness
  // зчитуємо і виводимо час hh:mm:ss через buf
  for (uint8_t i = 0; i < 8; i++){
    getHour();
  }
  // вихід прокручуванням зображення догори
  for (uint8_t i = 0; i < 8; i++){
    mx.transform(MD_MAX72XX::TSU);
    delay(WAIT);
  }
  // зчитуємо і виводимо дату hh:mm через buf
  getDate();
  printText(0, MAX_DEVICES-1, buf);
  scrollText(buf);
  getWeekDay();
  scrollText(buf);
  getTemp();
  printText(0, MAX_DEVICES-1, buf);
  delay(20*WAIT);
  // Time setting in RTC:
  if (Serial.available() > 0 && timeset == false) {
    adjustClock(Serial.readString());
  }
  if (timeset == true)
  {
     detachInterrupt(digitalPinToInterrupt(SET_BTN));
     mx.clear();
     adjustTimeDate();
     mx.clear();
     timeset = false;
     attachInterrupt(digitalPinToInterrupt(SET_BTN), set_time_date_ISR, CHANGE);
  }
}
```

Додаток 2

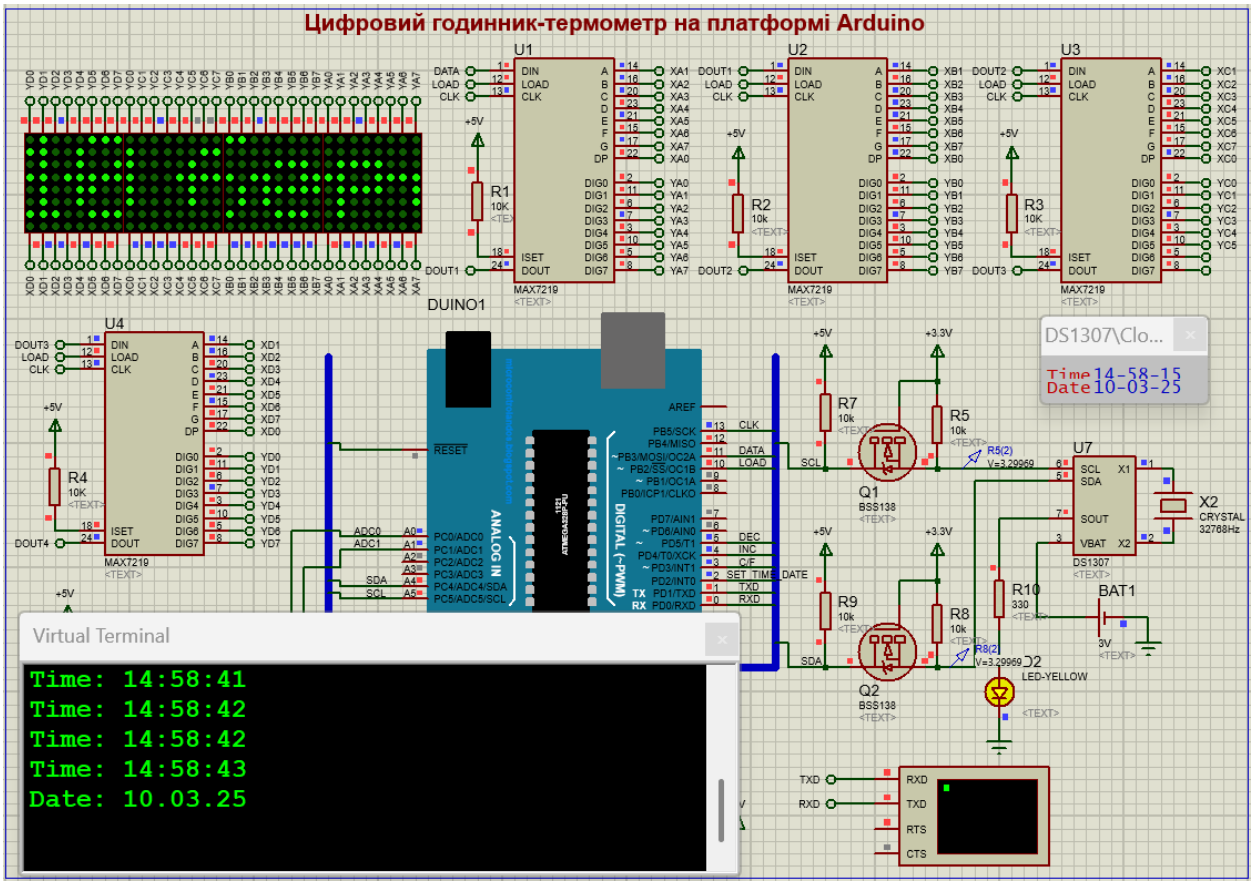
Моделювання цифрового годинника-термометра в Proteus ISIS.

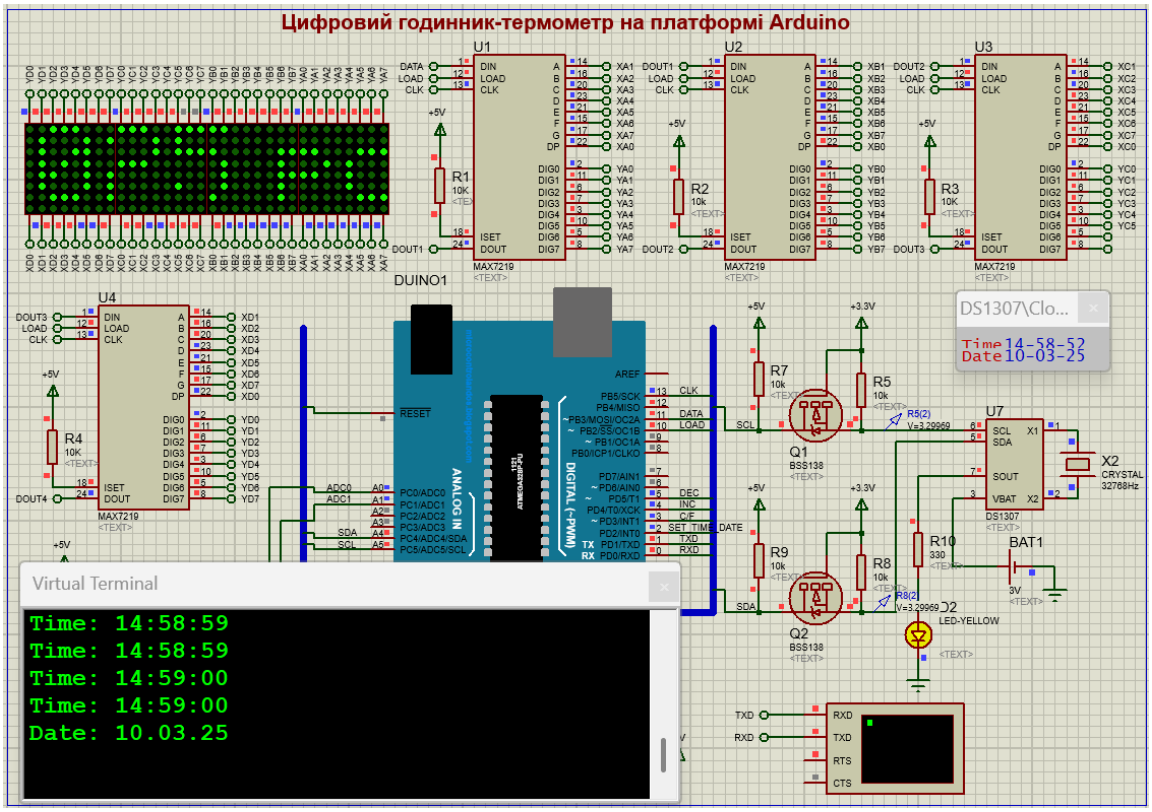


Вивід інформації про пристрій та його розробника

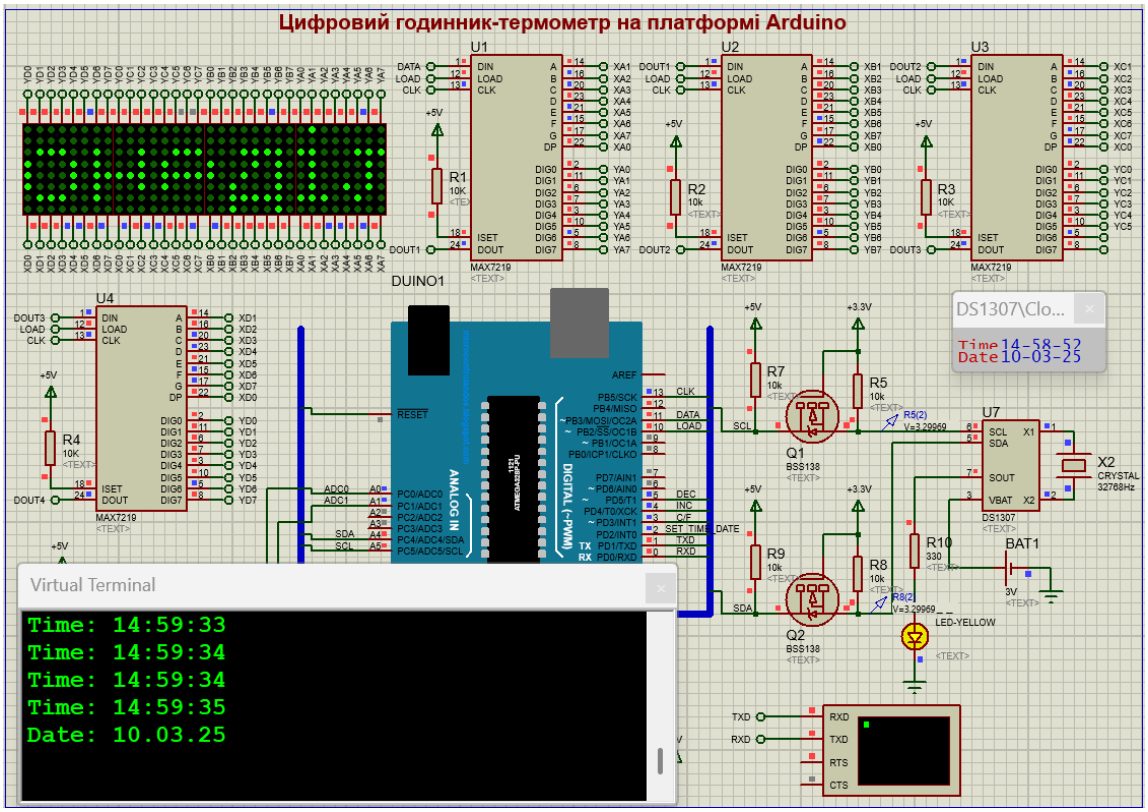


Вивід поточного часу

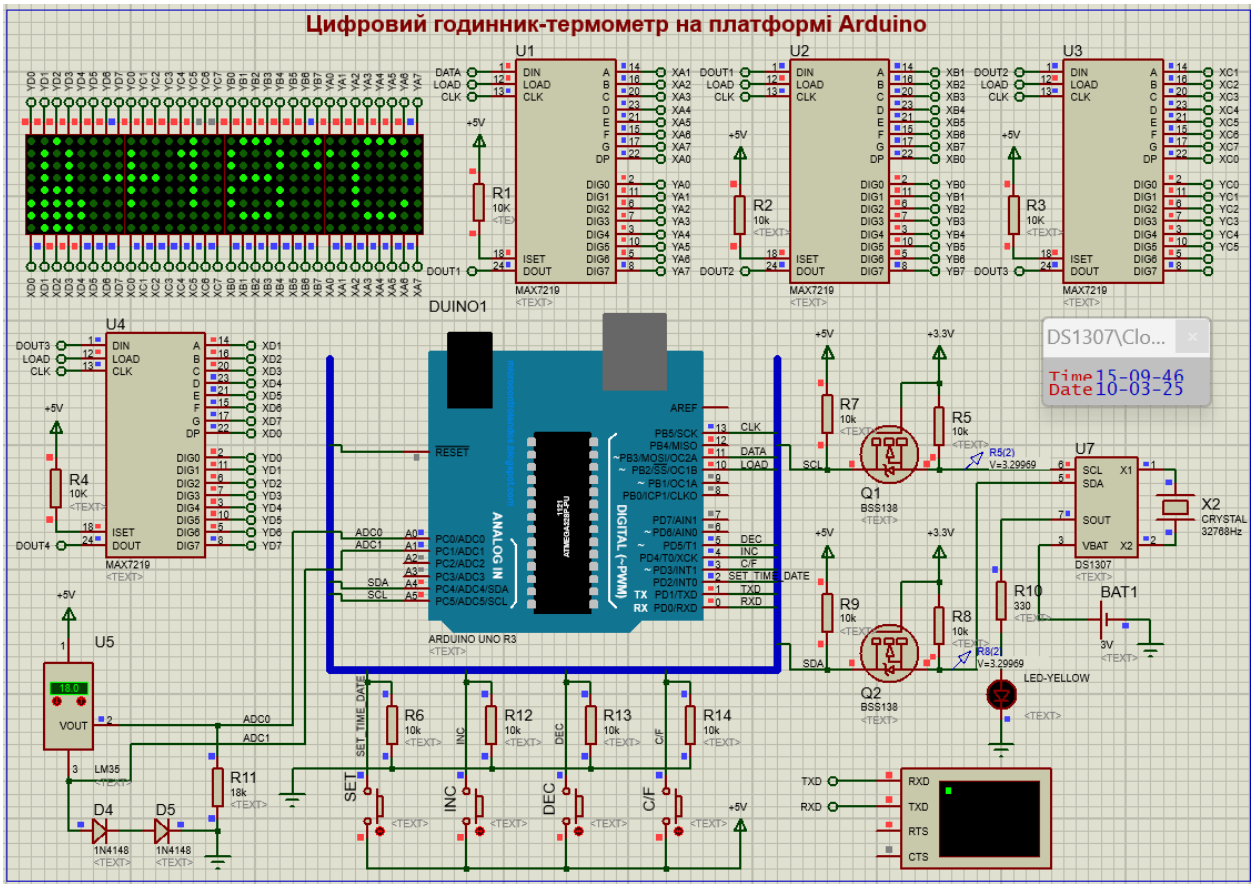


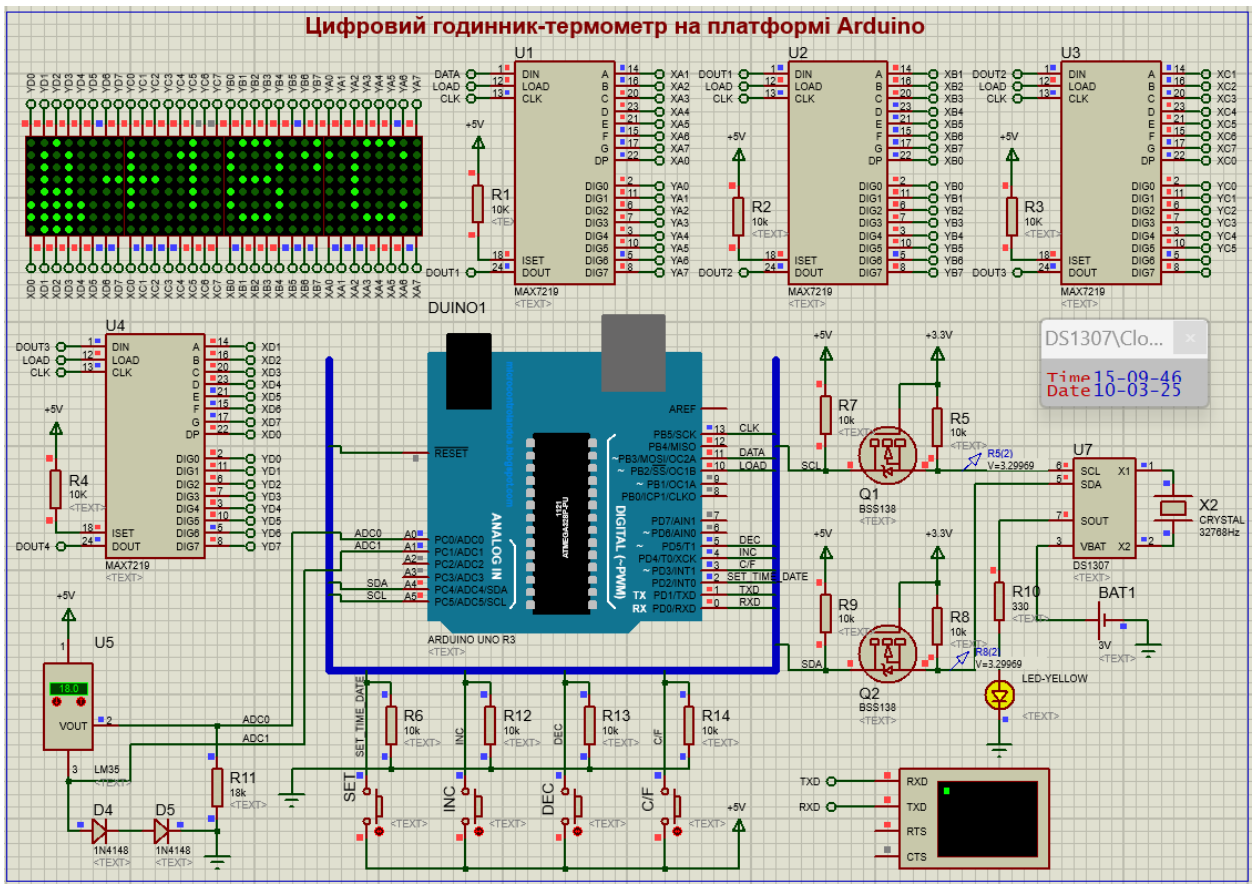


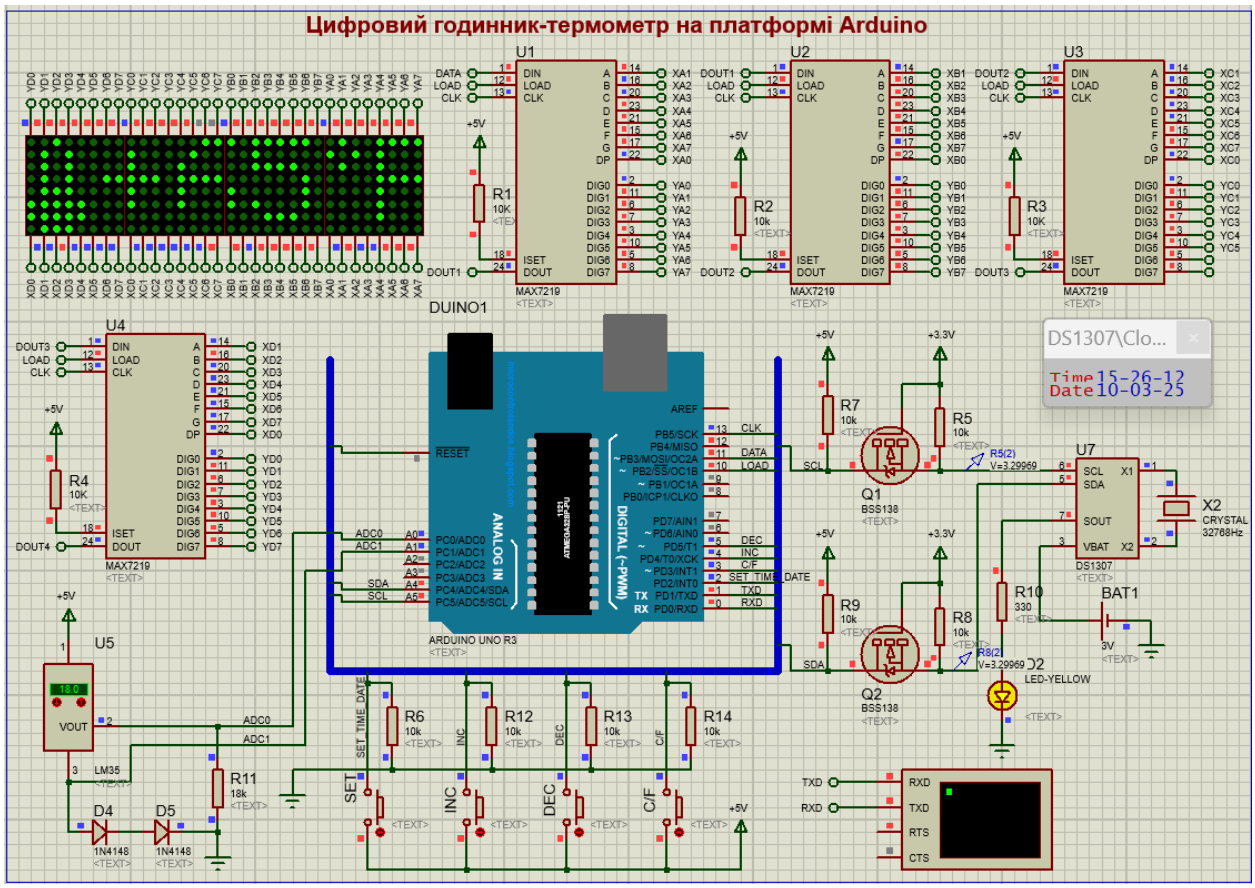
Вивід поточної дати



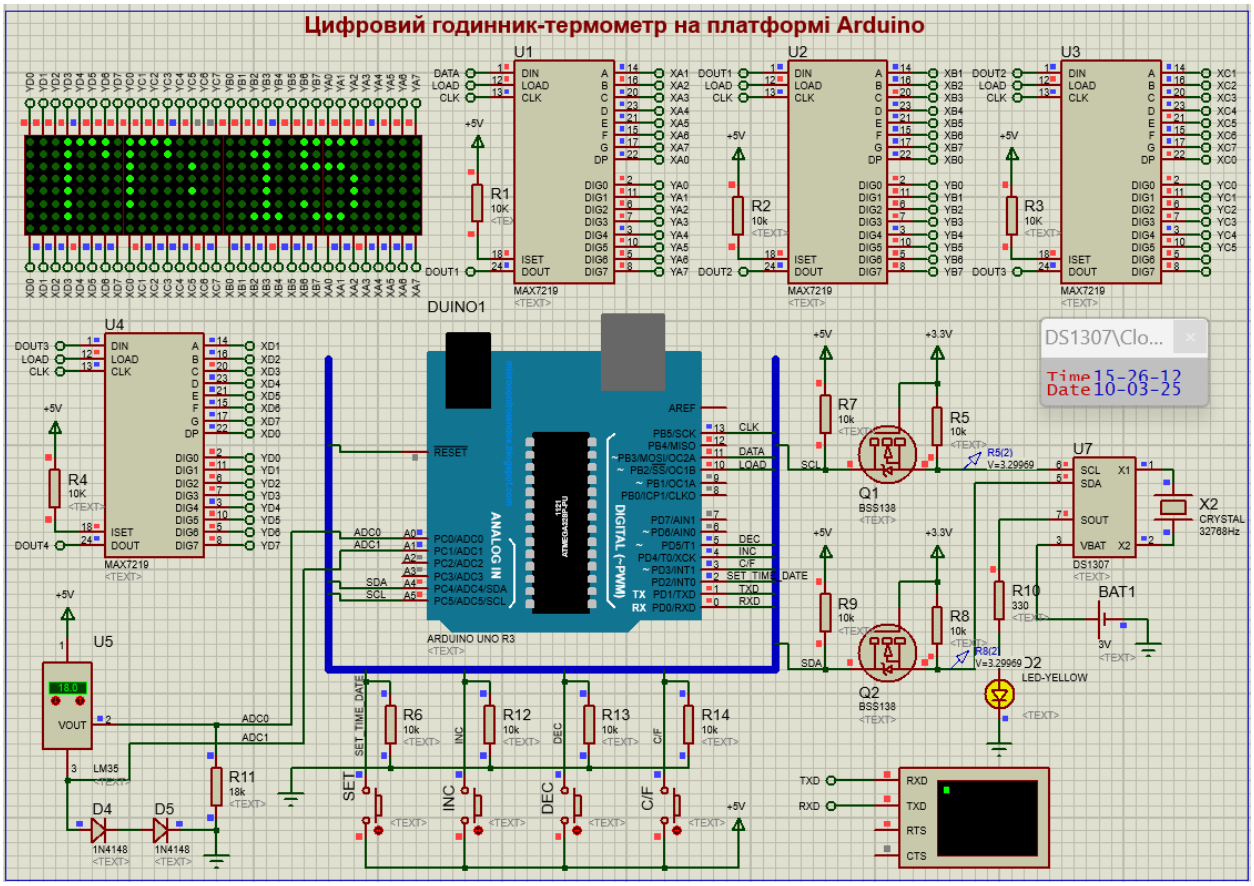
Вивід дня тижня

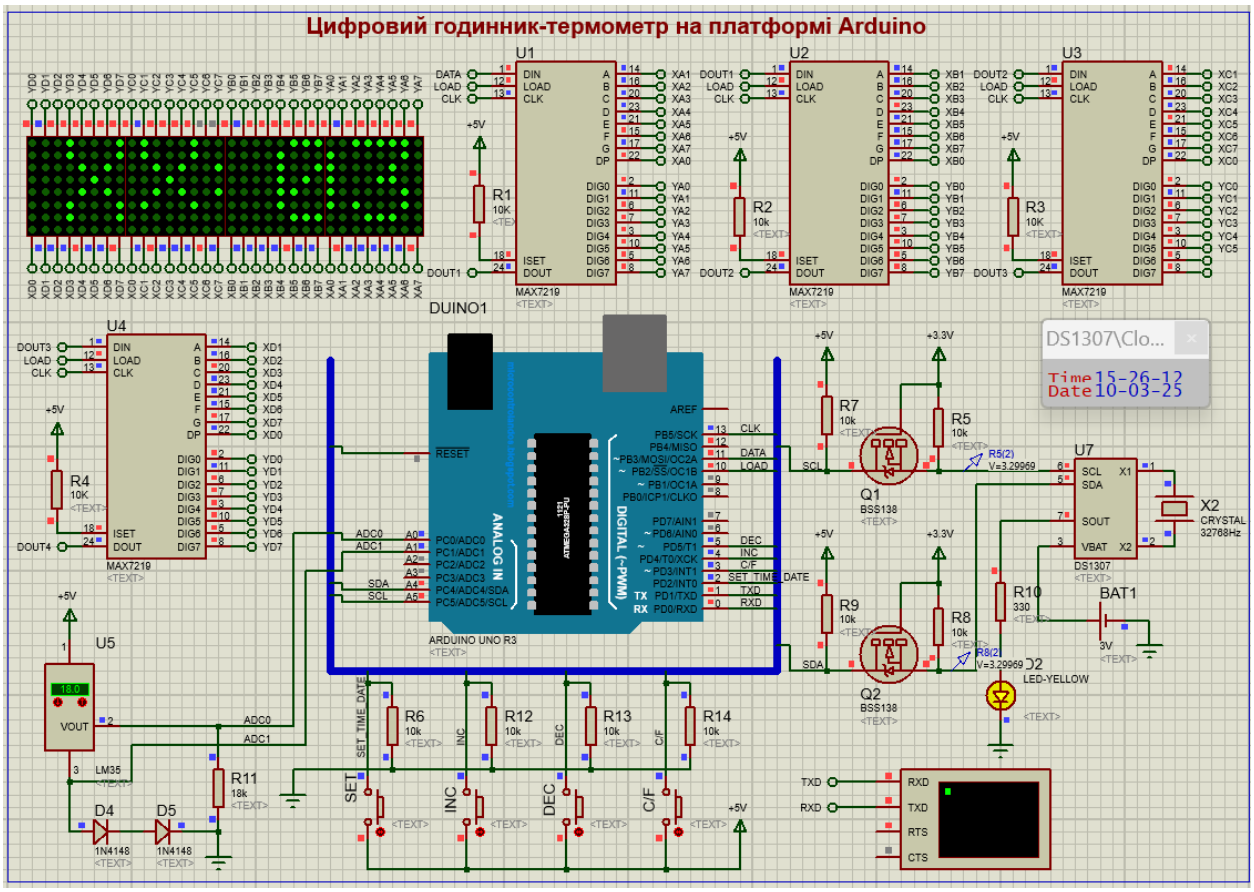


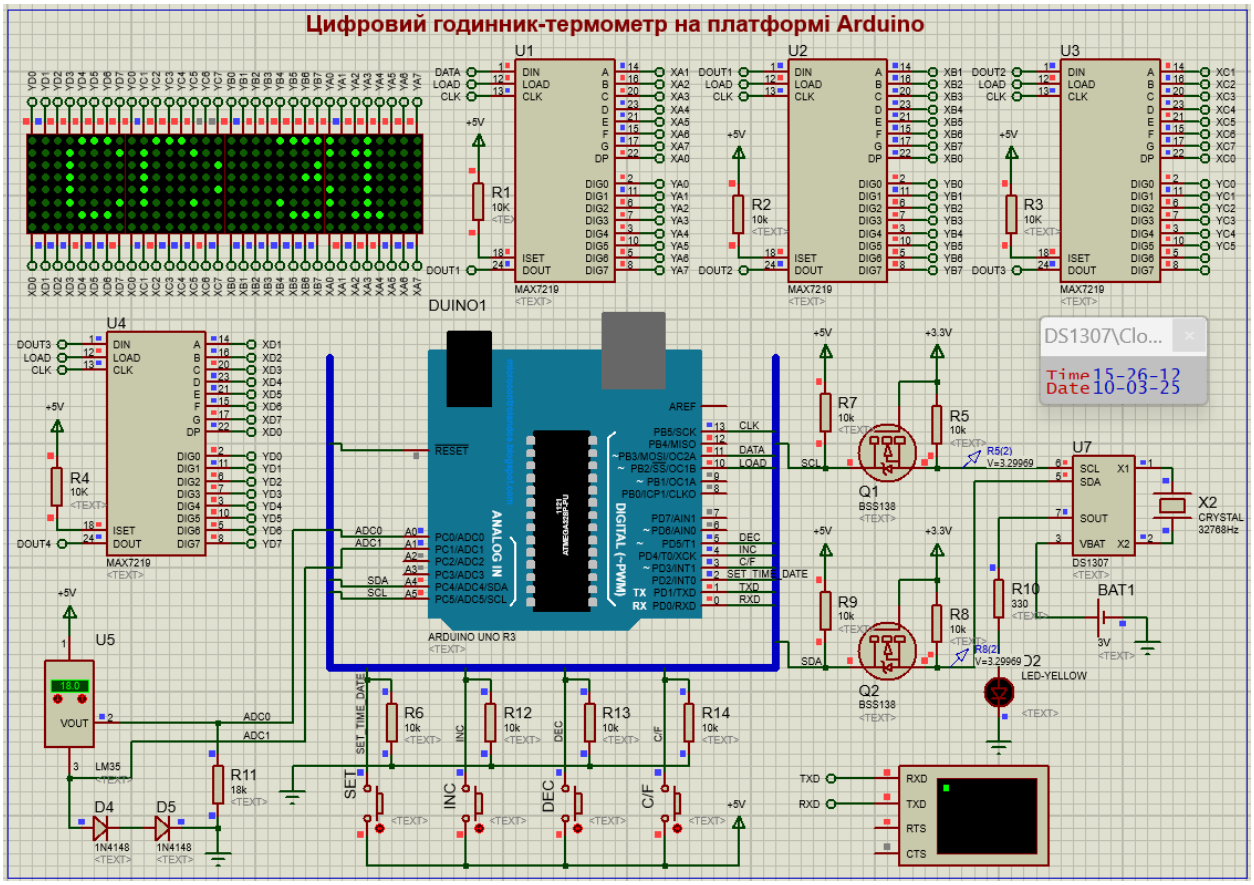




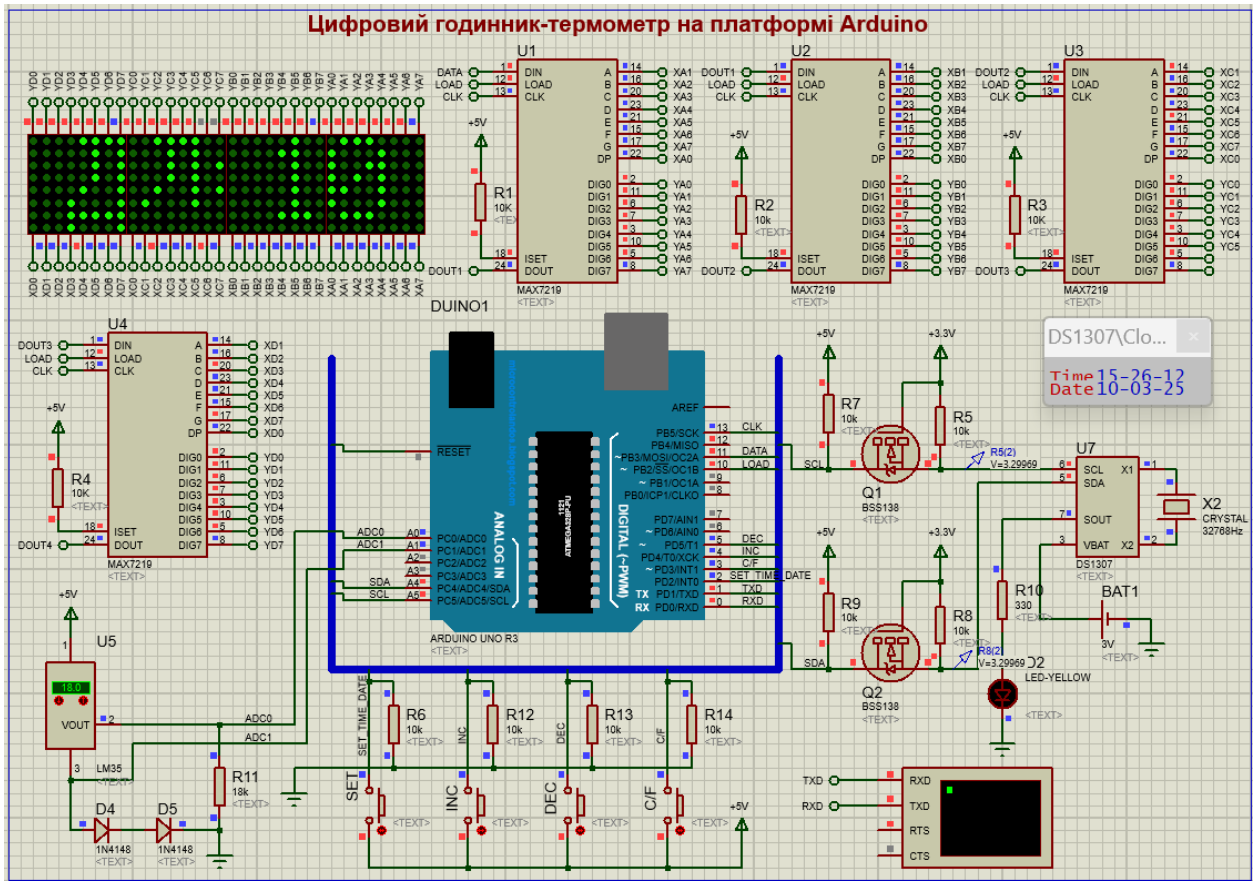
Моделювання роботи цифрового годинника-термометра. Вивід температури

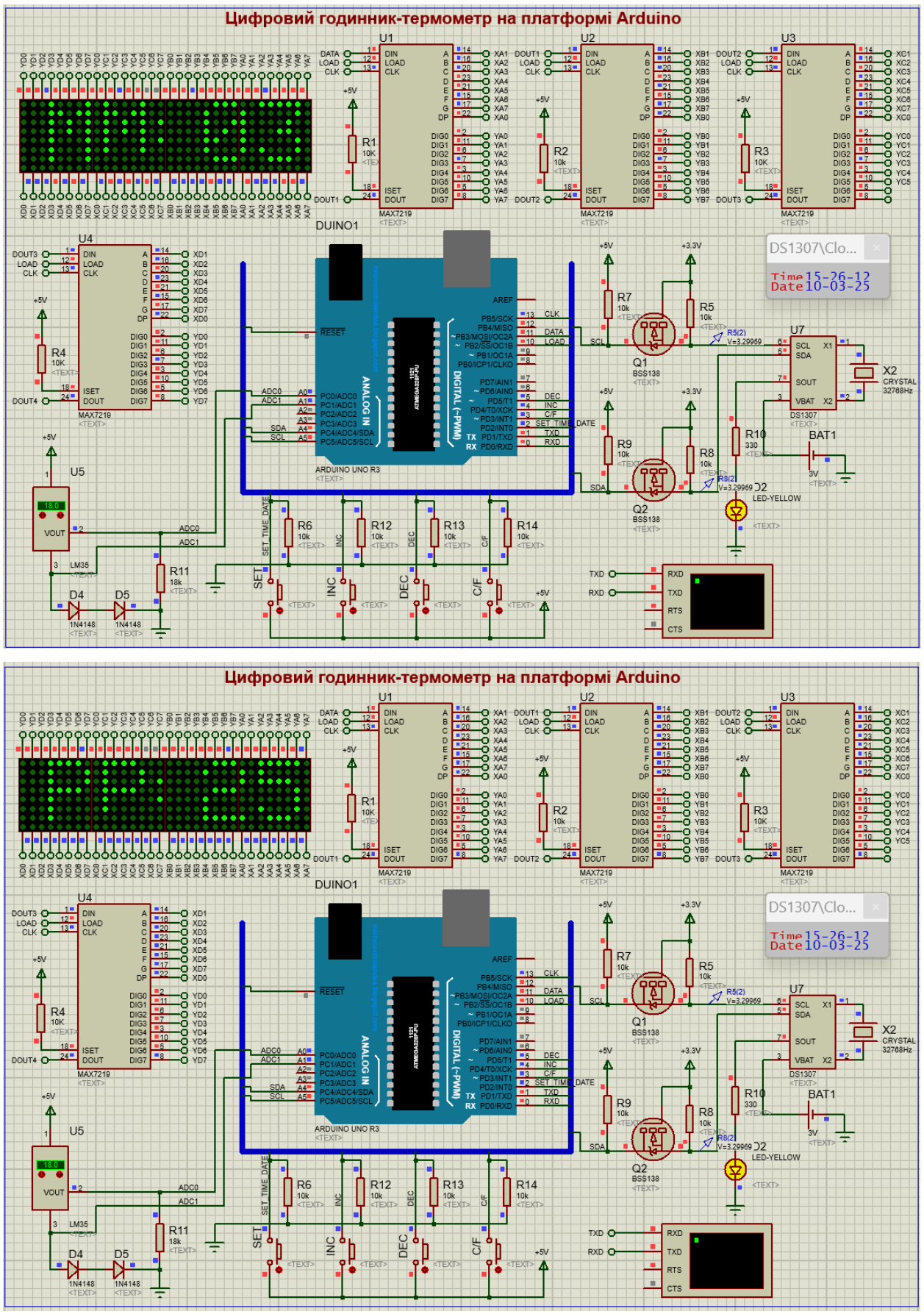

Цифровий годинник-термометр на платформі Arduino
U1
U2
U3
U4
U5
U7
DUINO1
MAX7219
ARDUINO UNO R3
DS1307\Clo...
Time 15-26-12
Date 10-03-25
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
Q1
Q2
BSS138
X2
CRYSTAL
32768Hz
BAT1
3V
D2
LED-YELLOW
D4
D5
1N4148
LM35
SET
INC
DEC
C/F
SET TIME DATE
RXD
TXD
RTS
CTS


Цифровий годинник-термометр на платформі Arduino
U1
U2
U3
U4
U5
U7
DUINO1
MAX7219
ARDUINO UNO R3
DS1307\Clo...
Time 15-26-12
Date 10-03-25
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
Q1
Q2
BSS138
X2
CRYSTAL
32768Hz
BAT1
3V
D2
LED-YELLOW
D4
D5
1N4148
LM35
SET
INC
DEC
C/F
SET TIME DATE
RXD
TXD
RTS
CTS



Моделювання роботи цифрового годинника-термометра. Встановлення часу





Моделювання роботи цифрового годинника-термометра. Встановлення дати

Додаток 3

Дослідження макету цифрового годинника-термометра