МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

КВАЛІФІКАЦІЙНА РОБОТА

за освітнім ступенем «магістр»

на тему: Гаджет часу на основі Arduino

Виконав: магістрант 2 курсу
групи М-КН-21
спеціальності 122 «Комп'ютерні науки»
Зінчук Ярослав Степанович
Керівник: к.ф.-м.н. Ляшук Т.Г.

Рівне – 2023

РЕФЕРАТ

У кваліфікаційній роботі представлено процес розробки розумного годинника на основі мікроконтролера Arduino.

Об'єкт дослідження – смарт-годинники, мікроконтролер Arduino Uno та його периферія (Годинник реального часу RTC DS1302, LCD 1602 I2C дисплей, датчик вологості та температури DHT11).

Предмет дослідження – історія, особливості і функціонал смарт-годинники; аналіз технічних характеристик Arduino Uno, тестування взаємодії з іншими компонентами; розгляд структури та компонентів RTC DS1302, аналіз технічних параметрів; аналіз технічних характеристик LCD 1602 I2C, вивчення принципу роботи; аналіз технічних параметрів DHT11, особливості роботи датчика при вимірюванні вологості та температури.

Була побудована схема та створений на її основі розумний годинник. Як основа використовувалася мікроконтролерна плата Arduino Uno. Також для реалізації деякого функціоналу використовувались: датчик температури та вологості DHT11 і годинник реального часу RTC DS1302. Для виводу інформації використовується LCD 1602 I2C. Для реалізації функціоналу смарт-годинника був розроблений скрипт і було проведено тестування його роботоздатності.

Кваліфікаційна робота складається зі змісту, вступу, трьох розділів та висновків до кожного з них, загальних висновків та списку використаних літературних джерел із 32 найменувань. До роботи додається реферат українською та англійською мовами, а також додатки. Загальний обсяг роботи складає 76 сторінок.

Ключові слова: смарт-годинник, мікроконтролер, Arduino Uno.

ABSTRACT

The qualifying paper presents the process of developing a smart watch based on an Arduino microcontroller.

The research object is a smart watch, Arduino Uno microcontroller and its peripherals (RTC DS1302 real-time clock, LCD 1602 I2C display, humidity and temperature sensor DHT11).

The subject of the study is the history, features and functionality of smart watches; analysis of technical characteristics of Arduino Uno, testing interaction with other components; consideration of the structure and components of RTC DS1302, analysis of technical parameters; analysis of technical characteristics of LCD 1602 I2C, study of the principle of operation; analysis of technical parameters of DHT11, features of sensor operation when measuring humidity and temperature.

A circuit was built and a smart watch was created based on it. The Arduino Uno microcontroller board was used as the basis. Also, to implement some functionality, the temperature and humidity sensor DHT11 and the RTC DS1302 real-time clock were used. LCD 1602 I2C is used to display information. To implement the functionality of the smart watch, a script was developed and its functionality was tested.

The qualification work consists of a table of contents, an introduction, three sections and conclusions to each of them, general conclusions and a list of 32 items of used literary sources. The work is accompanied by an abstract in Ukrainian and English, as well as appendices. The total volume of work is 76 pages.

Keywords: smart watch, microcontroller, Arduino Uno.

Зміст

ВСТУП

Актуальність теми. Смарт-годинники широко використовують в різних сферах, так і у побуті. Можливість створити розумний годинник на основі мікроконтролерної плати Arduino наддасть ентузіастам деякі переваги:

- *Модульність та гнучкість*. З'явиться можливість змінювати функціонал годинника за рахунок зміни скрипта, датчиків та модулів.
- *Низька вартість.* Arduino – платформа відома своєю доступністю, та низькою ціною.
- *Доступність ресурсів*. Широкий вибір бібліотек та модулів для розширення функціоналу годинника.

Мета: Розробка смарт-годинника на основі мікроконтролеру Arduino, який поєднує в собі традиційні функції годинника з передовими можливостями сучасної технології.

Для досягнення поставленої мети були поставленні наступні **завдання**:

- *Підбір компонентів*: Вибір необхідних електронних компонентів, таких як мікроконтролер, LCD дисплей, датчик температури та вологості, годинник реального часу та інші.
- *Розробка апаратної частини*: Збірка та підключення обраної апаратної оснастки для годинника, враховуючи енергоефективність та зручність.
- *Програмування*: Написання програмного забезпечення, яке включає в себе основні функції годинника, а також додаткові можливості, такі як вимірювання вологості та температури повітря, зберігання поточного часу попри знеструмлення годинника.
- *Тестування та відладка*: Проведення випробувань для перевірки працездатності та надійності годинника, виявлення та виправлення можливих помилок.

Об'єкт дослідження – (внутрішня/зовнішня) периферія мікроконтролера.

Предмет дослідження – апаратно-програмна складова смарт-годинника.

Методи дослідження: Експериментальний метод передбачає собою проведення тестів та випробувань для оцінки функціональності та надійності годинника. Аналіз технічних можливостей включає вивчення ресурсів Arduino та електронних

компонентів, та їх програмний супровід.

Наукова новизна отриманих результатів: Створений смарт-годинник дає можливість не тільки користуватися початково створеним функціоналом, а й доповнювати його за рахунок додаткових модулів, які сумісні з Arduino Uno, і можливості розширення даного скрипта.

Апробація результатів дослідження: Основні положення та результатів дослідження доповідались та обговорювались на:

- XV Всеукраїнської науково-практичної конференції «Інформаційні технології в професійній діяльності» (Рівне, 1 листопада 2022 р.);
- XVI Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих учених «Наука, освіта, суспільство очима молодих» (Рівне, 19 травня 2023 р.);

Структура роботи: Кваліфікаційна робота складається зі змісту, вступу, трьох розділів, висновків та списку використаних літературних джерел. До роботи додається реферат українською та англійською мовами.

Загальний обсяг роботи складає 76 сторінок. Робота містить 13 рисунків, 14 таблиць. Список використаних літературних джерел складає 23 найменування.

У першому розділі «Смарт-годинник. Історія їх виникнення та розвитку» викладена коротка інформація про смарт-годинник та їх функціонал та особливості, описана історія їх виникнення та розвиток до сьогодення. Другий розділ «Об’єкти та методи досліджень» містить у собі опис всіх необхідних модулів для створення розумного годинника на основі Arduino, їх характеристики, побудову та особливості. У третьому розділі «Апаратно-програмна реалізація» описана апаратна та програмна реалізація проекту, а саме схема годинника, поетапне підключення модулів, опис функціоналу та його програмний опис.

РОЗДІЛ 1 СМАРТ-ГОДИННИК. ІСТОРІЯ ЇХ ВИНИКНЕННЯ ТА РОЗВИТКУ

1.1 Смарт-годинники. Їх особливості

Смарт-годинник **[1, 2]** – це електронний пристрій, який окрім основної функції показу поточного часу володіє також додатковими: погода, біоритм та ряд інших. Найпопулярніші додаткові функції старт-годинників, які реалізуються їх виробниками, можна виділити наступні [3]:

- отримання повідомлень (GSM, соціальні мережі), дзвінків, електронних листів;
- моніторинг стану здоров’я свого власника (серцебиття, тиск тощо);
- GPS-навігація, яка дозволяє визначати місцезнаходження користувача;
- керування іншими смарт-приладами.

На період 2023 року до типових особливостей розумних годинників можна віднести:

- здатність підключатися до смартфона: смарт-годинники можуть взаємодіяти зі смартфонами через Bluetooth. Це дозволяє отримувати сповіщення та приймати дзвінки;
- вимірювання фізичних параметрів, спортивні та фітнес-функції: багато сучасних смарт-годинників мають сенсори для вимірювання серцевого ритму, водяного балансу, кількості спалених калорій, кількість пройдених кроків та відстань;
- наявність NFC: на даний момент більшість годинників обладнані технологією ближнього зв'язку (NFC), що дозволяє здійснювати безконтактні платежі;
- наявність GPS-навігації: більшість смарт-годинників мають вбудований GPS-модуль, який дозволяє знаходити місцезнаходження користувача.

1.2 Історія розвитку смарт-годинників

1.2.1 Виникнення та ранні роки

Перший електронний годинник з’явився у 1972 році і мав назву *Pulsar* [4, 5] (виробництва *Hamilton Watch Company)*. По функціоналу він міг показувати тільки

час, але в своїй конструкції він містив сенсор, який давав можливість налаштовувати яскравість екрану. Ціна такого годинника становила 2100 доларів, що було еквівалентно ціні автомобіля. Ціна була зумовлена тим, що корпус був зроблений з 18-каратного жовтого золота.

Рис. 1.2.1 Годинник *Pulsar P1*.

У 1982 році *Seiko* випускає годинник *Pulsar (NL C01)*. Цю модель можна вже по справжньому назвати смарт-годинником. Він був першим годинником із пам’яттю і вмів запам’ятовувати текст довжиною до 24 символів, що давало можливість для написання невеличких заміток.

Рис. 1.2.2 Годинник *Pulsar NL C01*.

У 1980-х роках все ті ж *Seiko* почали розробляти годинник з функціоналом комп’ютера і вже в кінці 1983 року з’являється модель *Seiko Data 2000* [6]. Годинник поставлявся з зовнішньою клавіатурою для введення даних. Клавіатура підключалася до годинника через бездротову док-станцію використовуючи для цього електромагнітний зв’язок. Годинник був здатний зберігати 2000 символів.

Рис. 1.2.3 Годинник *Seiko Data 2000*.

Також в 1984 році *Seiko* випускає модель *RC-1000* [7]. Це була перша модель *Seiko*, яка підключалася до комп'ютера. Він мав 2 Кб пам'яті та дворядковий 12-символьний дисплей. Передача даних відбувалася за допомогою інтерфейсу RS232C. Враховуючи те, що інтерфейс був сумісний з більшістю комп'ютерів того часу, це давало можливість підключати годинник майже до всіх персональних комп'ютерів (скорочено ПК) того часу. Ціна годинника становила 100 фунтів стерлінгів.

Рис. 1.2.4 Годинник *Seiko RC-1000* підключений до ПК.

1.2.2 Розвиток смарт-годинників у 1990-ті та 2000-ні рр.

У 1994 році *Timex Datalink Smartwatches* випускає годинник *Timex Datalink* [8]. Ранні годинники Timex Datalink реалізували режим бездротової передачі даних із ПК. Заплановані зустрічі та контакти, створені за допомогою Microsoft Schedule+, попередника MS Outlook, можна було легко передавати на годинник за допомогою бездротовій мережі.

Рис. 1.2.5 Годинник *Timex Datalink* .

У 1998 році Стів Манн створив прототип першого у світі наручного годинника з підтримкою операційної системи Linux, який він представив на IEEE ISSCC2000 7 лютого 2000 року.

Рис. 1.2.6 *Прототип годинника на операційній системі Linux* на обкладинці *Linux Journal.*

Тим часом у 1998 році *Seiko* випустили в Японії *Ruputer* [9] – комп'ютер-наручний годинник із процесором 3,6 МГц. Модель не була сильно успішною, оскільки замість сенсорного екрану вона використовував маніпулятор, схожий на джойстик, для введення символів, а маленький екран із роздільною здатністю 102x64 у 4 градації сірого ускладнював читання великої кількості символів.

Рис. 1.2.7 Годинник *Seiko Ruputer* .

За межами Японії цей годинник відомий як *Matsucom onHand PC*. Незважаючи на досить низький попит, *Matsucom onHand PC* поширювався і підтримувався до 2006 року, що робило його смарт-годинником з досить тривалим життєвим циклом.

У 1999 році *Samsung* випустила перший у світі телефон-годинник SPH-WP10 [10]. Він мав виступаючу антену, монохромний РК-екран і 90 хвилин роботи в режимі розмови з вбудованим динаміком і мікрофоном.

Рис. 1.2.8 Годинник *Samsung SPH-WP10* .

У червні 2000 року *IBM* показала прототип *WatchPad* [11], наручного годинника, який працював під управлінням *Linux* 2.2. Оперативна пам’ять годинника складала 8 МБ. Початкова версія мала лише 6 годин роботи від батареї, яку пізніше було продовжено до 12. Пізніше пристрій було оновлено акселерометром, вібраційним механізмом і датчиком відбитків пальців.

WatchPad 1.5 мав монохромний сенсорний дисплей 320 × 240 QVGA і працював під управлінням *Linux* 2.4. Він також мав програмне забезпечення для календаря, Bluetooth, 8 МБ оперативної пам’яті та 16 МБ флеш-пам’яті. Годинник планувалось продавати за 400 доларів.

Рис. 1.2.9 Годинник *IBM WatchPad* .

Також у вересні 2000 році *Epson Seiko* представили свій наручний годинник *Chrono-bit*. Годинник мав безель, який використовувався для введення даних, синхронізовують дані PIM через послідовний кабель.

У 2003 році *Fossil* випустила *Wrist PDA* [12, 13]. Це годинник, який працював під управлінням *Palm OS* і містив 8 МБ оперативної пам’яті та 4 МБ флеш-пам’яті. Також він мав у своєму розпорядженні вбудований стилус для використання крихітного монохромного дисплея з роздільною здатністю 160×160 пікселів. На той час багато рецензентів оголосили годинник революційним, але розкритикували за вагу у 108 грамів. Цей годинник зняли з виробництва в 2005 році.

Рис. 1.2.10 Годинник Fossil Wrist PDA.

У тому ж році *Microsoft* анонсувала смарт-годинник *Smart Personal Objects Technology* [14, 15] (скорочено *SPOT*), а надходити в магазини він почав на початку 2004 року. Пристрій був смарт-годинником, який надавав таку інформацію як: новини, погоду, ціни на акції та результати спортивних змагань. Передавалась інформація через хвилі FM. Годинник мав монохромний екран 90×126 пікселів. *Fossil, Suunto* і *Tissot* також продавали смарт-годинники з технологією *SPOT*.

Рис. 1.2.11 Годинник Microsoft SPOT Watch .

У 2006 році *Sony Ericsson* об’єдналася з *Fossils* і випустила перший годинник *MBW-100* [16, 17], який підключався через *Bluetooth* до телефона. Цей годинник сповіщав користувача про отримання дзвінків і текстових повідомлень. Сам годинник не набув великої популярності через особливість підключення до телефону, оскільки міг працювати лише з телефонами *Sony Ericsson*.

Рис. 1.2.12 Годинник *Sony Ericsson MBW-100* .

У 2009 році Гермен ван ден Бург, генеральний директор *Smartwatch* і *Burg Wearables*, випустив *Burg* – перший окремий годинник для смартфона з власною SIM-картою, який не потребує прив'язки до смартфону.

Також у цьому році, *Samsung* випустила телефон-годинник *S9110* [18], який мав 1,76-дюймовий кольоровий РК-дисплей з товщиною 11,98 мм.

Рис. 1.2.13 Годинник *Samsung* S9110.

1.2.2 Розвиток смарт-годинників у 2010-ті та 2020-ні рр.

28 вересня 2010 року *Sony Ericsson* запускає *Sony Ericsson LiveView* [19], переносний годинник, який по суті є зовнішнім *Bluetooth* дисплеєм для смартфона *Android*.

Рис. 1.2.14 Годинник *Sony Ericsson LiveView*.

В 2011 році *Vyzin Electronics Private Limited* випустили смарт-годинник із *ZigBee*

Remote Health Monitoring [20] та стільниковим зв’язком для віддаленого моніторингу здоров’я під назвою *VESAG*.

Рис. 1.2.15 Годинник *ZigBee Remote Health Monitoring*.

У 2011 році *Motorola* випускає *MOTOACTV 6.* Годинник містить програми для моніторингу спортивної активності за допомогою вбудованих акселерометра та GPS для вимірювання кроків і відстані відповідно. Його також можна синхронізовувати з велосипедним датчиком швидкості або крокоміром за допомогою технології ANT+. Годинник може спілкуватися із зовнішніми пристроями через Bluetooth 4.0, такими як датчик пульсу та стереонавушники. Крім того, є режим DJ, який динамічно адаптує музику до тренування. Також ця модель містить в собі графічний процесор PowerVR, 256 МБ оперативної пам’яті та 8 ГБ або 16 ГБ флеш-пам’яті NAND. Motorola припинила виробництво годинника в 2013 році.

Рис. 1.2.16 Годинник Motorola MOTOACTV 6.

В січні 2013 року компанія *Pebble* почала постачати годинники *Pebble* [21] спонсорам Kickstarter. Фінансування на розробку цього смарт-годинника було зібрано з допомогою пожертвувань людей з Kickstarter. Годинник мав 32-мм (144 × 168) пк чорно-білий РК-дисплей з пам’яттю, вібраційним двигуном, магнітометром, датчиками зовнішнього освітлення та тривісним акселерометром. Він може спілкуватися з пристроєм Android або iOS за допомогою Bluetooth 2.1 і Bluetooth 4.0. Годинник заряджається через модифікований USB-кабель, який магнітно приєднується до годинника, щоб зберегти водонепроникність. Батарея працює на протязі семи днів.

Рис. 1.2.17 Годинник Pebble .

У 2014 році *Omate* випустили смарт-годинник *TrueSmart* [22]. Це був перший годинник, який прив’язувався до всіх можливих смартфонів. Годинник мав TFT дисплей, (240 × 240) пікселів, двоядерний процесор ARM Cortex-A7 – 1.3 GHz, пам’ять 4 GB, операційну систему Android 4.2, Omate UI 2.0, батарю 600 мАг.

Рис. 1.2.18 Годинник Omate TrueSmart .

Також станом на 4 вересня 2013 року було випущено три нових розумних годинники: *Samsung Galaxy Gear, Sony SmartWatch 2* і *Qualcomm Toq*.

У квітні 2014 року *Samsung Gear 2* [23] був випущений серед небагатьох розумних годинників, оснащених цифровою камерою. Він має роздільну здатність 2 Мп з функцією запису відео у 720p. В порівнянні з *Galaxy Gear*, найбільш суттєвою зміною в лінійці *Gear 2* стала заміна *Android* на розроблену *Samsung* операційну систему *Tizen* , що обіцяла покращену функціональність і час автономної роботи.

Рис. 1.2.19 Годинник Samsung Gear 2.

На виставці споживчої електроніки «Consumer Electronics Show 2014» було випущено велику кількість нових розумних годинників від різних компаній, зокрема як *Razer Inc* та *Archos*, а також кількох стартапів. Дехто почав називати 2014 рік «роком революції на зап'ясті» через велику кількість випущених розумних годинників і рекламну компанію, яку вони почали отримувати на початку 2014 року.

В серпні 2014 року відбувся запуск смарт-годинника *Gear S* від *Samsung*. Модель мала вигнутий дисплей *Super AMOLED* і вбудований 3G модем. Корпорація почала продавати смарт-годинник *Gear S* у жовтні 2014 року разом із аксесуаром для гарнітури *Gear Circle*.

Рис. 1.2.20 Годинник *Samsung Gear S*.

На виставці IFA 2014 *Sony Mobile* анонсувала третє покоління серії розумних годинників *Sony Smartwatch 3* на базі Android Wear. Крім того, було анонсовано електронний паперовий годинник Fashion Entertainments.

На початку 2015 року *Apple Inc*. випустила свій перший розумний годинник під назвою *Apple Watch* [24]. Перша спроба *Apple* впровадити технологію переносних пристроїв була зустрінута значною критикою в період перед запуском, оскільки в багатьох ранніх оглядах, користувачі зсилались на проблеми годинників з часом роботи батареї та несправністю апаратного забезпечення. Однак інші хвалили *Apple* за створення потенційно модного пристрою. Годинник вмикається, лише у випадку його активації (піднімаючи зап'ястя, торкаючись екрана чи натискаючи кнопку).

Рис. 1.2.21 Годинник Apple Watch.

29 жовтня 2014 року корпорація Майкрософт анонсувала *Microsoft Band*, розумний фітнес-трекер і перше розробка компанії в галузі пристроїв для носіння на зап'ясті після SPOT (технології інтелектуальних персональних об'єктів) десять років тому. *Microsoft Band* був випущений за ціною 199 доларів наступного дня, 30 жовтня 2014 року.

Рис. 1.2.22 Фітнес-трекер Microsoft Band.

У жовтні 2015 року *Samsung* представила *Samsung Gear S2*. Годинник працював під управлінням операційної системи Tizen, дисплей круглий Super AMOLED (360x360), пам'ять 4 ГБ, ОЗУ 512 МБ.

Рис. 1.2.23 Годинник Samsung Gear S2.

На виставці «Consumer Electronics Show 2016» *Razer* випустили *Nabu Watch* [25], розумний годинник із двома екранами. Перший екран містить постійно ввімкнений дисплей із підсвічуванням і обслуговує такі стандартні функції, як дата й час. Другий OLED-екран, який активується підняттям зап’ястя, надає доступ до додаткових інтелектуальних функцій.

Рис. 1.2.24 Годинник Razer Nabu Watch.

31 серпня 2016 року *Samsung* представили смарт-годинник *Samsung Gear S3* із кращими характеристиками. Існує щонайменше дві моделі: *Samsung Gear S3 Classic* і LTE-версія *Samsung Gear S3 Frontier*.

Рис. 1.2.25 Годинник Samsung Gear S3.

22 вересня 2017 року *Apple* випустила свою модель *Apple Watch Series 3*, яка пропонує вбудований стільниковий зв’язок LTE, що дозволяє здійснювати телефонні дзвінки, обмінюватися повідомленнями та передавати дані, не покладаючись на з’єднання поблизу смартфона.

Рис. 1.2.26 Годинник Apple Watch Series 3.

В 2018 році *Samsung* представила серію годинників *Samsung Galaxy Watch*.

Годинник мав розміри 42 мм (модель чорного і рожево-золотого кольору) і 46 мм (модель зі срібла). Дисплей з роздільною здатністю (360 x 360) пікселів, двоядерний процесор Exynos 9110 1,15 ГГц, акумулятор на 270 мАг і 472 мАг. Годинник містив в собі такі датчики, як акселерометр MEMS, MEMS гіроскоп, MEMS барометр, електрооптичний датчик (для моніторингу пульсу), фотодетектор (для зовнішнього освітлення).

Рис. 1.2.27 Годинник Samsung Galaxy Watch.

У вересні 2018 року *Apple* представила оновлений дизайн *Apple Watch Series 4*. Він мав більший дисплей із меншими рамками, а також функцію EKG, яка створена для виявлення аномальної серцевої діяльності.

Рис. 1.2.28 Годинник Apple Watch Series 4.

Qualcomm у вересні 2018 року презентував свій чіп *Snapdragon 3100*. Він є наступником *Wear 2100* і включає більшу енергоефективність і окреме ядро з низьким енергоспоживанням, яке може запускати базові функції годинника, а також більш розширені функції, такі як відстеження кроків.

В 2020 році Управління з санітарного нагляду за якістю харчових продуктів і медикаментів США схвалило маркетинговий додаток для *Apple Watch* під назвою *NightWare*. Додаток спрямований на покращення сну для людей, які страждають від кошмарів, пов’язаних із посттравматичним стресовим розладом, за допомогою вібрації, коли він виявляє кошмар, що триває, на основі моніторингу пульсу та рухів тіла.

Висновки

У даному розділі було розглянуто історію смарт-годинників, від їх зародження і перших експериментів, до сучасної масової популярності. Починаючи з перших цифрових годинників, таких як *Pulsar* від *Hamilton Watch Company* в 1972 році, відбувалась поступова еволюція до годинників з внутрішньою пам’яттю та можливістю підключатися до перших ПК. Представником даних годинників був *Data 2000* від *Seiko* в 1980 році.

Продовжуючи впровадженням повноцінних операційних систем та веб технологій у 2000-х та популяризацією у 2010-х.

Також було розглянуто їх типові особливості на сьогоднішній день, такі як:

- здатність підключатися до смартфона;
- вимірювання фізичних параметрів, спортивні та фітнес-функції;
- наявність NFC;
- наявність GPS-навігації.

РОЗДІЛ 2 ОБ’ЄКТИ ТА МЕТОДИ ДОСЛІДЖЕНЬ

2.1 Arduino Uno: технічні характеристики та можливості.

Arduino Uno **[26, 27, 28]** – це плата з відкритим вихідним кодом на основі мікроконтролера ATmega328P, розроблена Arduino.cc і вперше випущена в 2010 році. До її складу входить: 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШІМ-виходи), 6 аналогових входів, кварцевий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) та кнопка скидання. Плату можна живити від USB-кабелю або роз’єму, з вхідною напругою від 7 до 20 вольт.

Рис. 2.1.1 Плата Arduino Uno R3 (ATmega328, CH340G).

Табл. 1 Технічні характеристики Arduino Uno [29]

Параметр	**Значення**
Мікроконтролер	ATmega328
Робоча напруга	5 В
Напруга живлення (рекомендований)	(7÷12) В

Продовження таблиці 1

Напруга живлення (граничне)	(6÷20) В
Цифрові входи / виходи	14 (з них 6 можуть використовуватися в якості ШІМ-виходів)
Аналогові входи	6
Максимальний струм одного піна	40 мА
Максимальний вихідний струм виводу 3.3V	50 мА
Flash-пам'ять	32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16 МГц

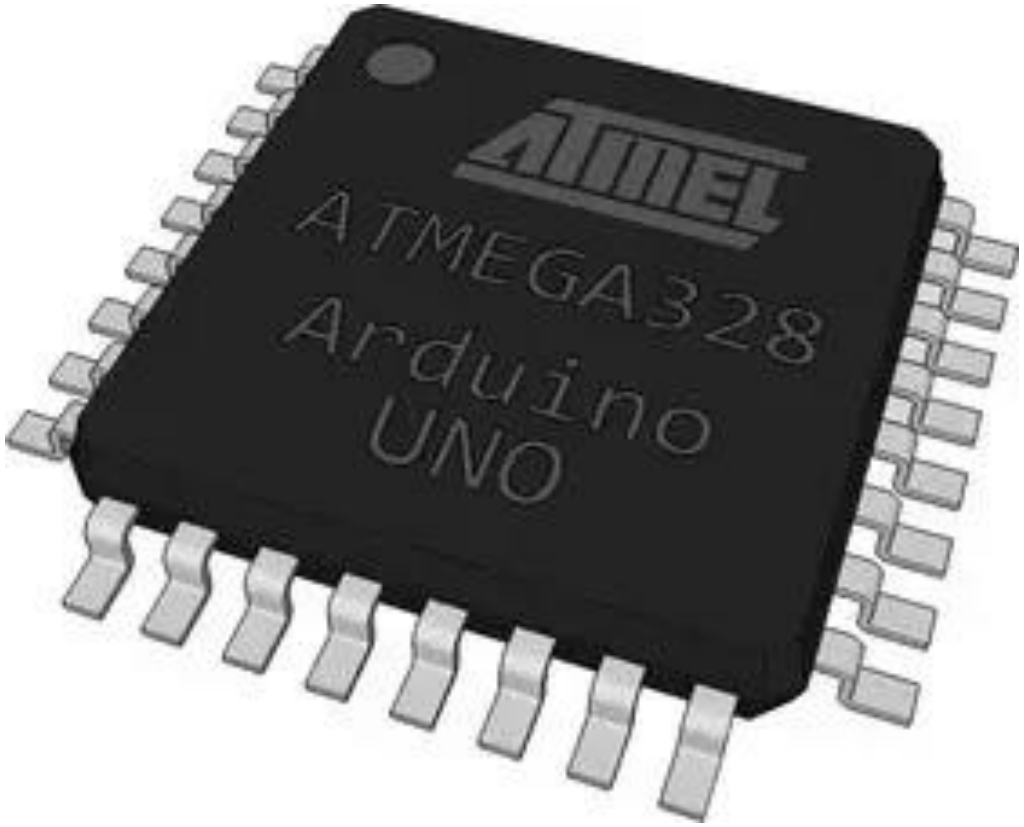


Рис. 2.1.2 Мікроконтролер ATmega328.

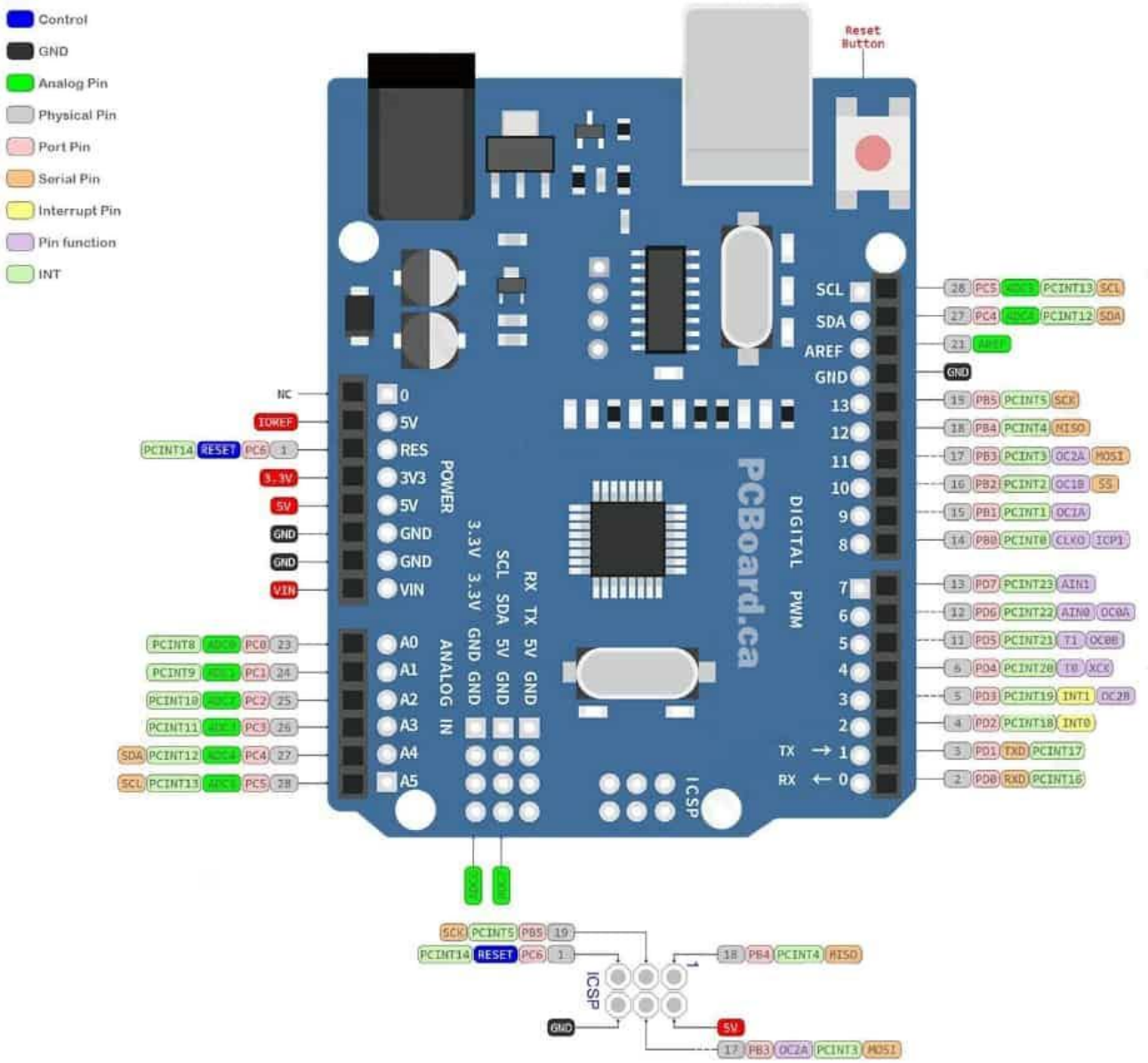


Рис. 2.1.3 Розпіновка плати Arduino Uno R3 (ATmega328, CH340G).

Загальні функції пінів **[29]:**

- *VIN*: Напруга, що надходить від зовнішнього джерела живлення (не пов'язана з 5 В від USB або іншого стабілізованою напругою). Через цей вивід можна як подавати живлення, так і споживати струм, у випадку коли пристрій живиться від зовнішнього джерела.
- *5V*: вихідна напруга 5V від регулятора на платі. Живлення плати може здійснюватися від роз’єму постійного струму (7÷20) В, роз’єму USB (5 В) або контакту VIN плати ((7÷20) В). Подача напруги через контакти 5V або 3.3V обходить регулятор і може пошкодити плату.
- *3.3V*: вихідна напруга 3,3 В, що створюється бортовим регулятором.

Максимальний споживчий струм становить 50 мА.

- *GND:* контакти заземлення.
- *IOREF*: Вивід надає платам розширення інформацію про робочу напругу мікроконтролера Arduino. Залежно від напруги, зчитаної з IOREF, плата розширення може перемкнутися на відповідне джерело живлення або задіяти перетворювачі рівнів, що дозволить їй працювати як з 5 В, так і з 3,3 В-пристроями.
- *RES*: зазвичай цей вивід служить для функціонування кнопки скидання на платах розширення.

Спеціальні функції пінів:

Кожен із 14 цифрових та 6 аналогових контактів на платі Arduino Uno може бути використаний як цифровий вхід або вихід. Робоча напруга пінів становить 5 В. Кожен з контактів може видавати струм 20 мА, що є рекомендованим стандартом, та має вбудований підтягуючий резистор у діапазоні від (20÷50) кОм. Важливо не перевищувати максимальне значення 40 мА на кожному вході/виході, щоб уникнути можливого негативного впливу на мікроконтролер. Arduino Uno також володіє 6-ма аналоговими входами, позначеними від A0 до A5, кожен з яких має роздільну здатність 10 біт. За замовчуванням, опорна напруга скидає до 5 В, проте її можна змінити за допомогою контакту AREF.

Крім того, деякі піни мають спеціальні функції:

- *UART*: контакти 0 (RX) і 1 (TX). Використовуються для отримання (RX) і передачі (TX) TTL послідовних даних. Такі контакти підключені до виводів контактів мікросхеми послідовного порту USB-TTL ATmega8U2;
- *зовнішні переривання*: контакти 2 і 3, які можна налаштувати для запуску переривання при низькому рівні, наростаючому або спадному фронті, або зміні рівні;
- *ШІМ (широтно-імпульсна модуляція)*: контакти 3, 5, 6, 9, 10 і 11. Може забезпечити 8-розрядний вихід ШІМ;
- *SPI*: контакти 10 (SS), 11 (MOSI), 12 (MISO) і 13 (SCK). Контакти підтримують інтерфейс передачі даних по протоколу послідовного периферійного

інтерфейсу;

- *TWI*: контакти SDA (A4) і SCL (A5), з підтримкою двопровідного послідовного інтерфейсу даних;
- *AREF*: опорна напруга для аналогових входів.

Зв'язок з комп'ютером.

Arduino Uno володіє низкою можливостей для взаємодії з комп'ютером, іншими платами Arduino або іншими мікроконтролерами. Мікроконтролер ATmega328 забезпечує послідовний зв'язок UART TTL (5 В), який доступний через цифрові контакти 0 (RX) і 1 (TX). Для реалізації такого послідовного зв'язку через порт USB використовується ATmega16U2, який виступає як віртуальний COM-порт для програмного забезпечення на комп'ютері. Прошивка 16U2 використовує стандартні драйвери USB COM, і додаткові зовнішні драйвери не є обов'язковими, за винятком файлу .inf, який потрібен для операційної системи Windows.

Також Arduino Uno має захист від перенавантаження USB-порту, який спрацьовує тоді, коли USB-порт споживається струм більший від 500 мА і не відновить з'єднання до усунення причини короткого замикання чи перевантаження.

Програмне забезпечення Arduino IDE включає послідовний монітор, який дозволяє обмінюватись даними між ПК і Arduino Uno. Світлодіоди RX і TX на платі вказують на передачу даних через чіп USB-послідовний та USB-з'єднання з комп'ютером (але не для послідовного зв'язку через контакти 0 і 1).

Для розширення можливостей послідовного зв'язку на платі Arduino Uno використовується бібліотека SoftwareSerial, яка надає можливість використовувати послідовний зв'язок на будь-якому з цифрових контактів.

Також конструкція Arduino Uno розроблена так, що замість фізичного натискання кнопки для скидання перед завантаженням скетча, вона дозволяє виконати скидання за допомогою програмного забезпечення, яке працює на підключеному комп'ютері. Одна з ліній управління потоком (DTR) ATmega8U2/16U2 з'єднана з лінією скидання ATmega328 за допомогою конденсатора 100 нФ. Це дає можливість при підключенні Arduino Uno до комп'ютерів, які працюють під управлінням Windows, Mac OS X або Linux, її мікроконтролер буде скидатися при

кожному з'єднанні програмного забезпечення із платою. Після скидання Arduino Uno на короткий час порядка (0,5 с) активується завантажувач, приблизно півсекунди. Навіть при тому, що завантажувач налаштований ігнорувати зайві дані (будь-які дані, які не відносяться до процесу прошивки новою програмою), він може перехопити перші кілька байтів інформації, що надходять на плату одразу після встановлення з'єднання. Тому, якщо програма на Arduino передбачає отримання від комп'ютера налаштувань або інших даних при першому запуску, необхідно переконатися, що програмне забезпечення, яке взаємодіє з Arduino, відправляє ці дані через секунду після встановлення з'єднання.

2.2 Годинник реального часу. RTC DS3231.

2.2.1 Загальні положення

Модуль RTC (годинник реального часу) – це електронний пристрій, який призначений для вимірювання і відстеження часу в реальному світі. Основна функція RTC полягає в тому, щоб забезпечити точний час для системи або пристрою, незалежно від того, чи вони вимкнені чи увімкнені.

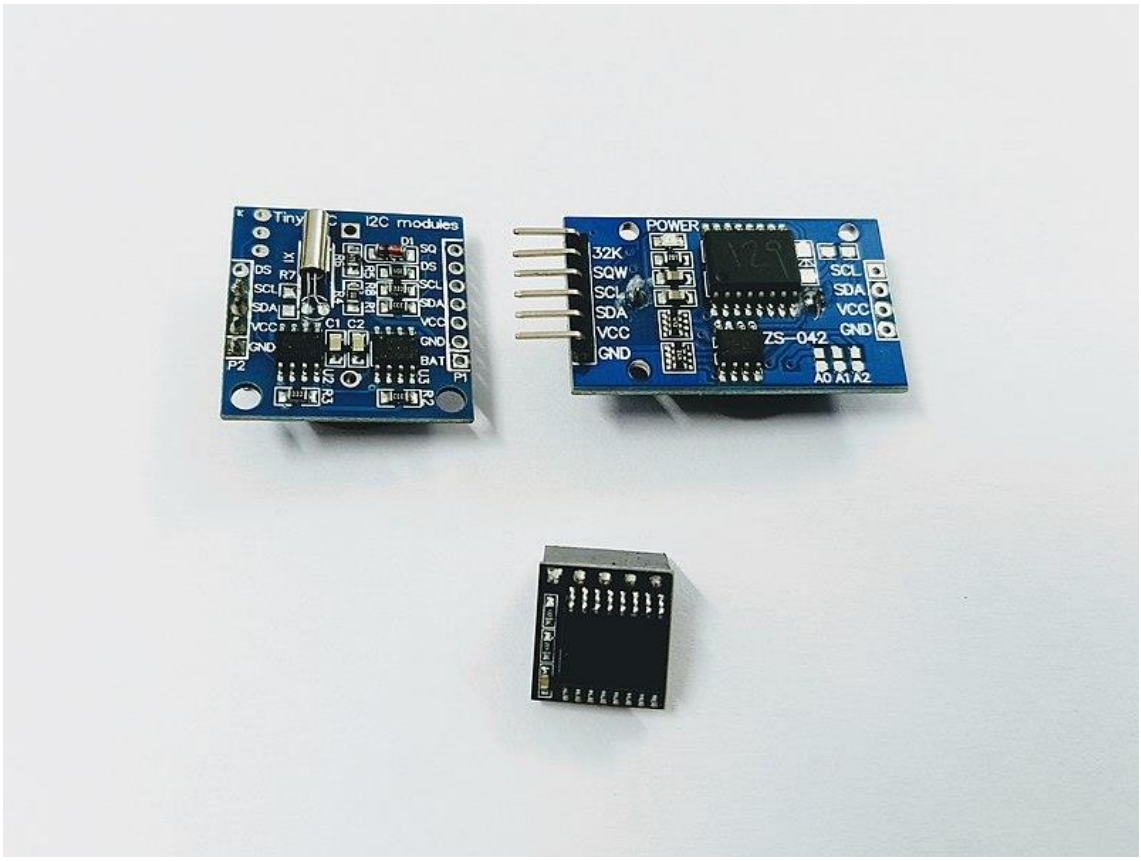


Рис. 2.2.1 Модулі RTC.

RTC може використовуватися в різноманітних пристроях, таких як комп'ютери, мікроконтролери, мікропроцесори, мобільні телефони, смарт-годинники та інші пристрої, де важлива точність часу.

Хоча можна вести облік часу і без RTC, його використання має свої переваги:

- понижене енергоспоживання;
- звільнення основної системи від виконання часово критичних завдань;
- надає більшу точність, ніж інші методи;
- GPS-приймач може прискорити час запуску, порівнюючи поточний час свого RTC із часом останнього дійсного сигналу.

RTC обладнані резервним джерелом енергії, що дозволяє їм продовжувати відлік часу навіть у тих випадках, коли основне джерело живлення вимкнене або недоступне. Таке додаткове джерело енергії може включати літієвий елемент живлення у старих системах, але у деяких нових системах використовується суперконденсатор, оскільки він може перезаряджатися та легко піддаватися пайці. Крім того, резервне джерело живлення може також живити оперативну пам'ять, що отримує енергію від акумулятора.

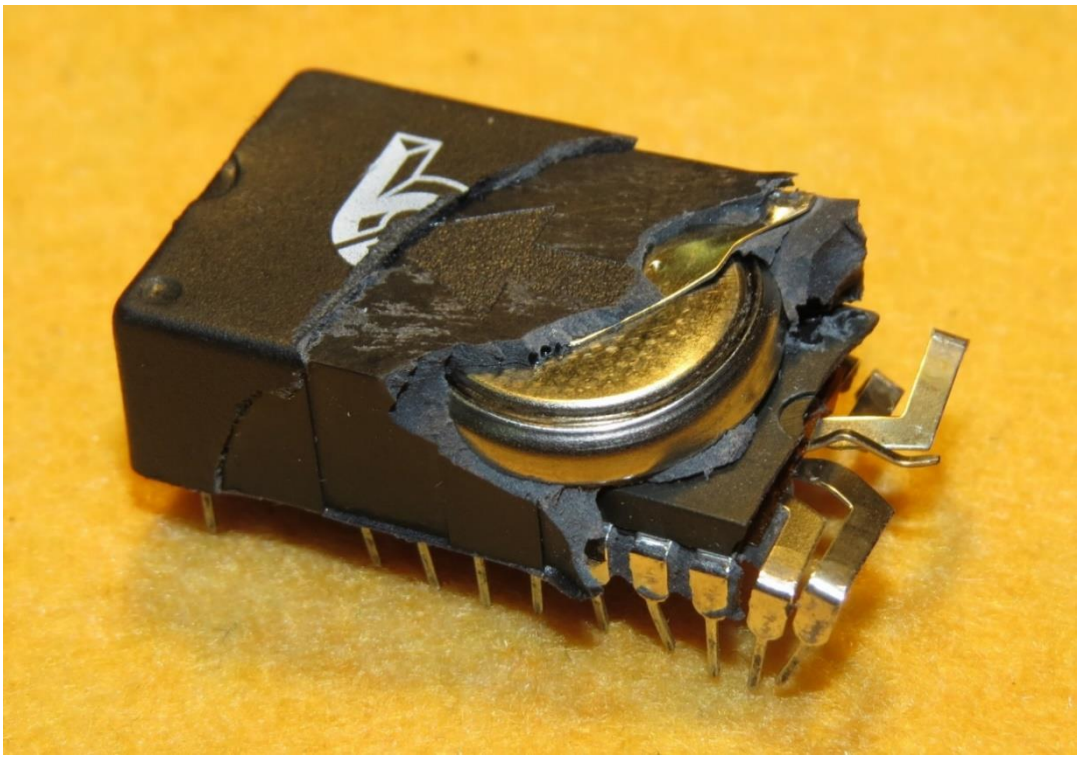


Рис. 2.2.2 Літієва батарея всередині мікросхеми RTC.

2.2.2 RTC DS1302. Архітектура та характеристики

DS1302 [30] - це інтегральна мікросхема годинника від DALLAS, американського виробника. Завдяки вбудованому годиннику/календарю реального часу та 31-байтовій статичній пам'яті вона може взаємодіяти з мікроконтролером через звичайний послідовний інтерфейс. Мікросхема надає інформацію про секунди, хвилини, години, день, тиждень, місяць і рік. DS1302 може автоматично коригувати кількість днів у місяці та у високосному році. Є можливість вибору між 24 або 12-годинною системою, обравши AM/PM.

Рис. 2.2.3 Годинник реального часу RTC DS1302.

Мікросхема взаємодіє з мікроконтролером за допомогою синхронного послідовного інтерфейсу і вимагає лише трьох інтерфейсних кабелів: кабелю скидання (RST), кабелю введення/виведення даних (SDA) і кабелю послідовного годинника (SCL).

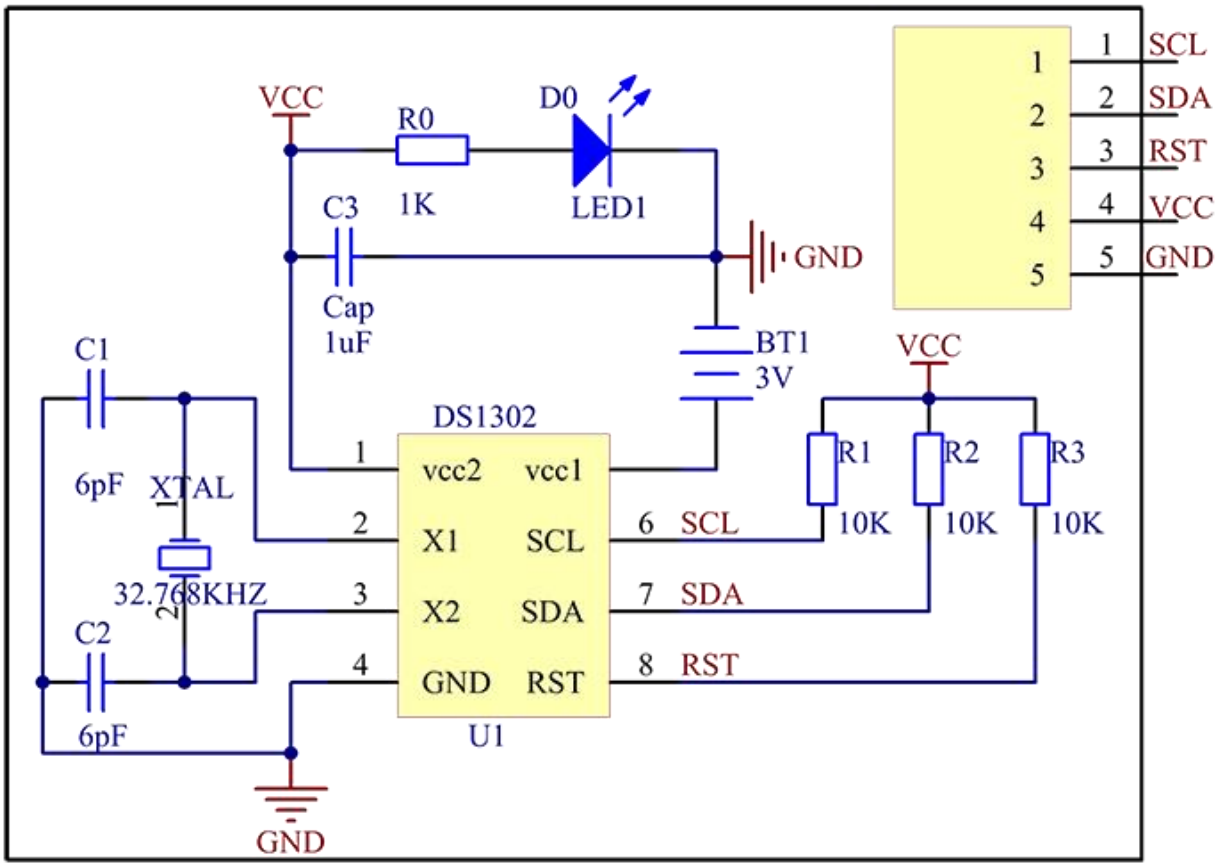


Рис. 2.2.4 Принципова схема модуля RTC DS1302.

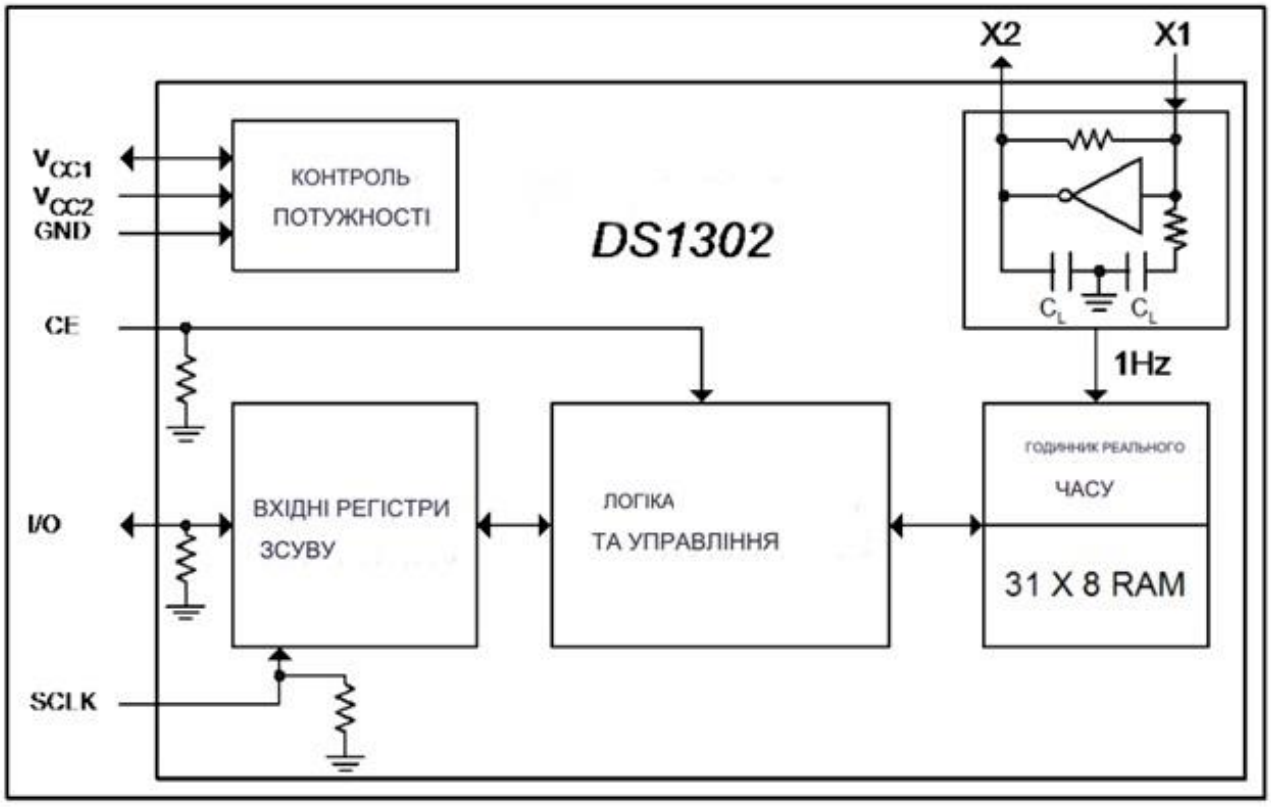


Рис. 2.2.5 Блок-схема модуля RTC DS1302.

Чіп DS1302 є покращеним аналогом DS1202. Основними відмінностями мікросхем є: додаткові 7 байт ОЗУ та два витоки для підключення живлення на платі з мікросхемою.

Функція пінів **[30]*:***

- V_{cc2} – основний контакт джерела живлення в конфігурації подвійного джерела живлення. V_{cc1} підключений до резервного джерела для підтримки часу та дати за відсутності основного живлення. DS1302 працює від контакту із більшою напругою. Коли V_{cc2} перевищує V_{cc1} + 0,2 В, V_{cc2} живить DS1302. При $V_{cc2} < V_{cc1}$, V_{cc1} живить DS1302.
- X1/ X2 – підключення при необхідності до зовнішнього кварцового кристала 32,768 кГц. В даній конфігурації пін X1 підключений до сигналу зовнішнього генератора.
- GND – земля.
- CE (RST) – вхідний пін, сигнал якого має бути високим під час читання або запису. Пін має внутрішній 40 кОм (типовий) резистор з підтяжкою на землю.
- I/O – двонаправлений пін вводу/виводу даних для 3-провідного інтерфейсу. Вивід має внутрішній резистор номіналом 40 кОм, що підтягується до землі.
- SCLK – вхідний пін для синхронізації руху даних по послідовному інтерфейсу. Вивід має внутрішній 40 кОм резистор, що підтягується до землі.
- V_{cc1} – контакт, який відповідає за роботу з низьким енергоспоживанням в системах з одним джерелом живлення та системами, що працюють від батареї, і резервне живлення від батареї малої потужності. У системах, які використовують зарядний пристрій для крапельної зарядки, джерело енергії, що перезаряджається, підключається до цього контакту.

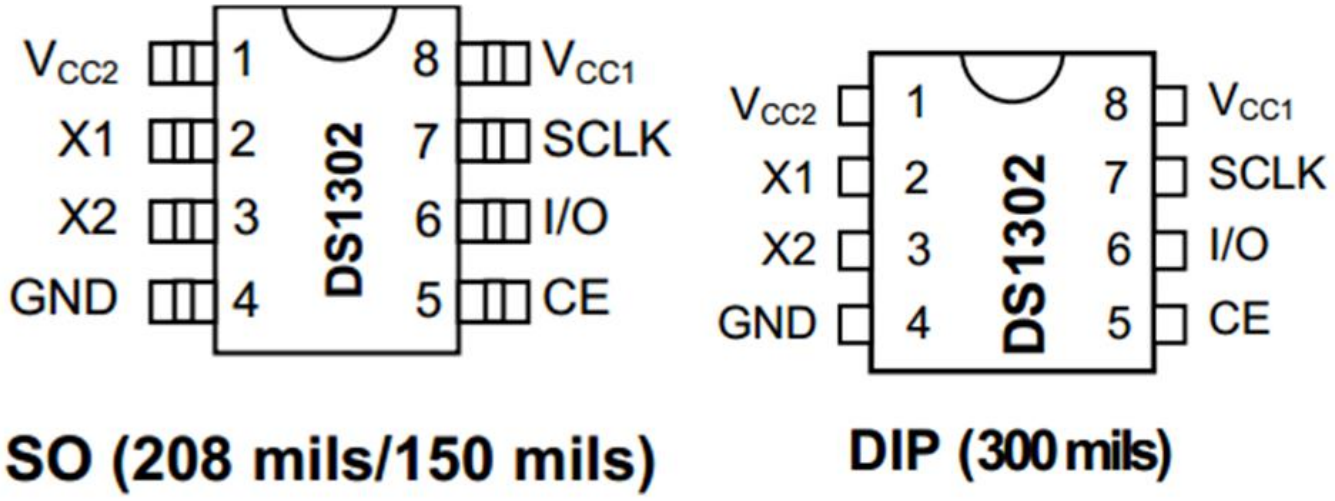


Рис. 2.2.6 Розпіновка мікросхеми DS1302

Табл. 2 Технічні характеристики модуля DS1302 RTC

Параметр	**Значення**
Робоча напруга	(2÷5,5) В
Елемент живлення	CR2032
Доступна пам'ять модуля	31 байт
Максимальний споживаний струм при 2,5 В	300 мА
Робоча температура	(-40÷85)°C
Розміри датчика	(44 x 23 x 12) мм

Додаткові особливості:

- повністю керує всіма функціями хронометражу;
- відлік часу дійсний до 2100 р.;
- прості інтерфейси послідовного порту для більшості мікроконтролерів;
- простий 3-провідний інтерфейс;
- TTL-сумісний (VCC = 5 В);
- однобайтова або багатобайтова (пакетний режим) передача даних для читання або запису даних годинника або оперативної пам'яті;
- робота з низьким енергоспоживанням подовжує час автономної роботи від батареї;
- 8-контактний DIP і 8-контактний SO мінімізують необхідний простір.

2.3 Рідкокристалічний дисплей LCD 1602 I2C: характеристики та особливості.

LCD1602 [31] – це свого роду матричний модуль для відображення літер, цифр, символів тощо. Він складається з позицій у матриці розміром 5x7 або 5x11; кожна позиція може відобразити один символ. Між двома символами розташована крапка, а між рядками – пробіл, розділяючи символи та рядки. Модель 1602 позначає, що вона може відображати 2 рядки по 16 символів.

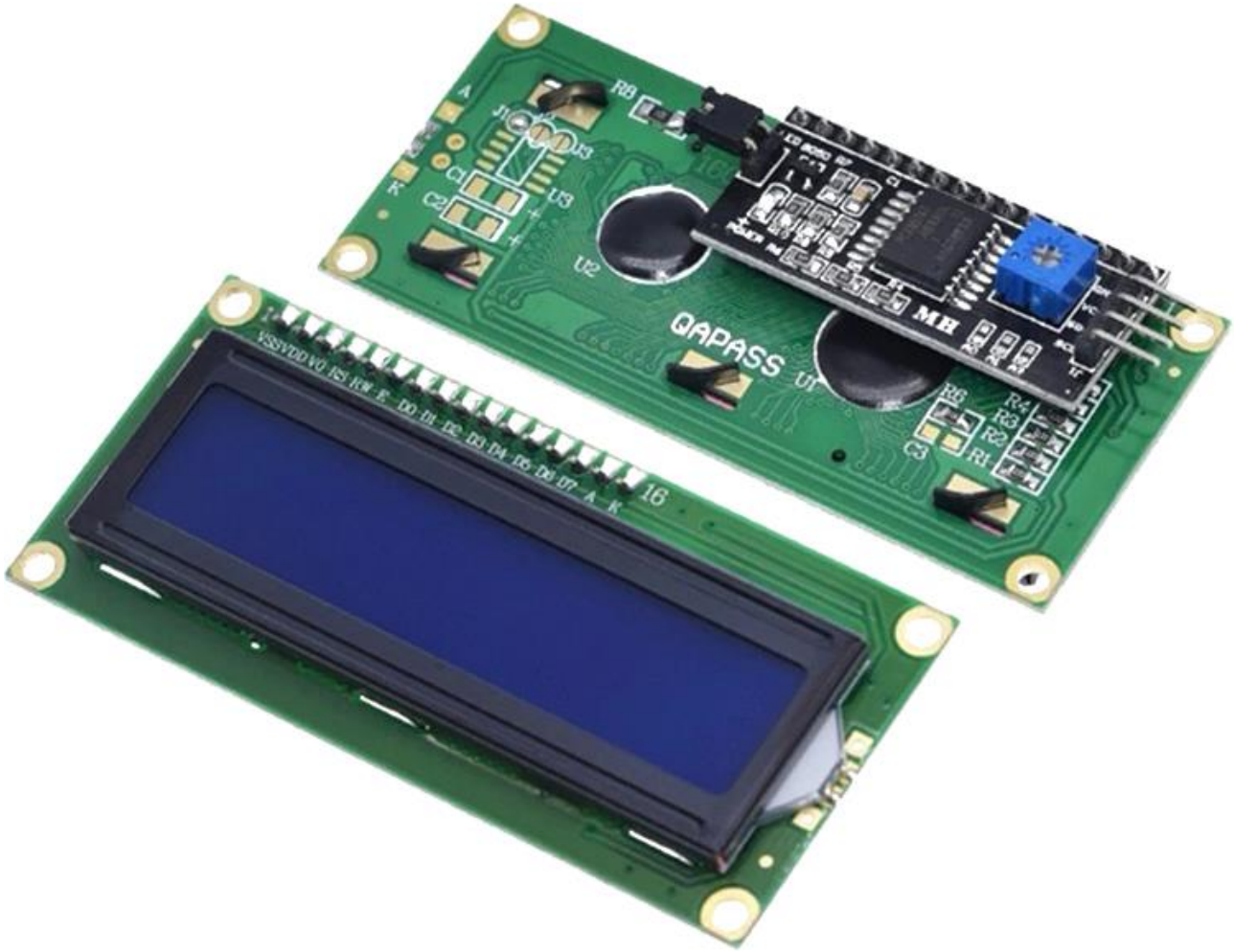


Рис. 2.3.1 Рідкокристалічний дисплей LCD 1602 I2C.

Зазвичай LCD1602 має паралельні порти, що означає, що він керує декількома контактами одночасно. Його підключення може бути восьмипортовим або чотирипортовим, проте для оптимального використання рекомендується використовувати чотирипортове підключення.

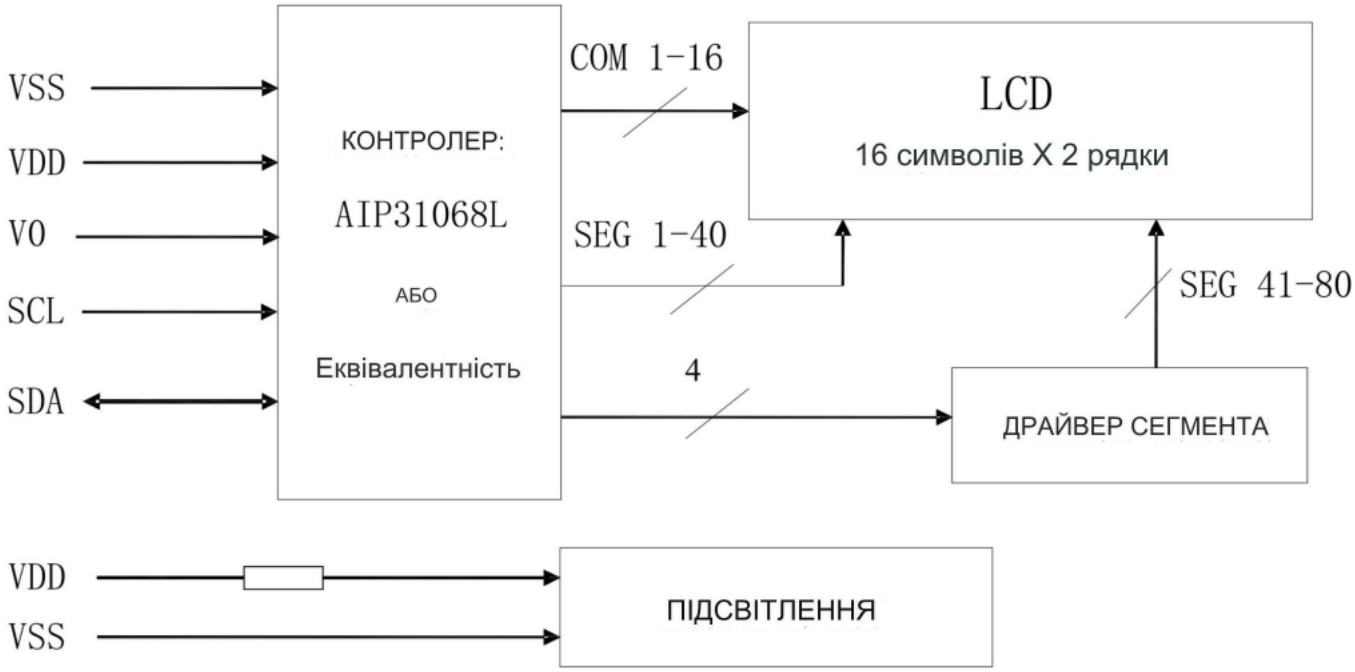


Рис. 2.3.2 Блок-схема модуля LCD 1602 I2C.

Функція пінів **[31]*:***

- VSS – земля;
- VDD – живлення +5 В;
- VO – регулювання яскравості дисплею;
- RS – пін вибору регістру, який контролює, куди в пам’ять РК-дисплея записуються дані;
- R/W – вибір між режимом читання та запису;
- E – пін активації, який зчитує інформацію, коли отримано високий рівень (1). Інструкції виконуються, коли сигнал змінюється з високого рівня на низький;
- D0-D7 – це піни для читання та запису даних;
- A – підсвічування дисплея. Підключається до 3,3 В;
- K – підсвічування дисплея. Підключається до землі.

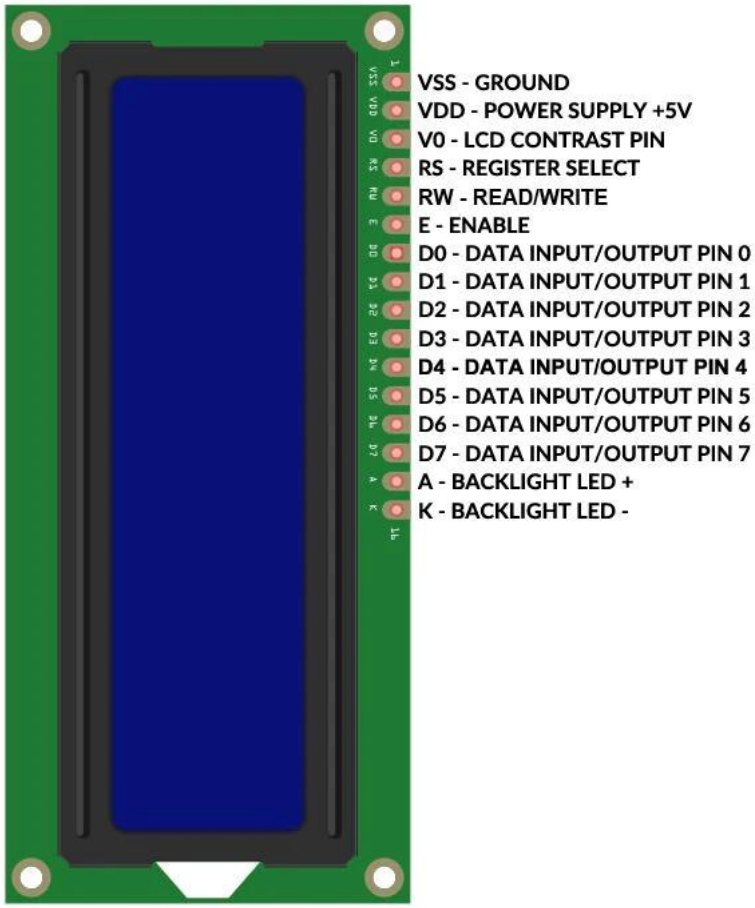


Рис. 2.3.3 Розпіновка модуля LCD 1602 I2C

При використанні протоколу I2C, кількість використовуваних пінів зменшується до чотирьох [31]:

- GND – земля.
- VCC – пін джерела живлення.
- SDA – пін даних.
- SCL – пін тактування.

Рис. 2.3.4 Розпіновка модуля LCD 1602 при використанні протоколу I2C

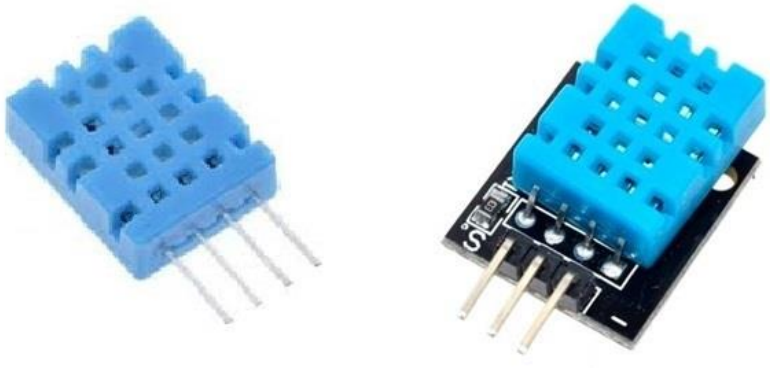
Технічні характеристики модуля LCD 1602 I2C **[31]*:***

- робоча напруга: (3,3 / 5) В;
- інтерфейс зв'язку: I2C;
- тип екрану: LCD;
- чіп управління: AiP31068L;
- розміри дисплея: (64,5 х 16,0) мм;
- габаритні розміри: (87,0 х 32,0 х 13,0) мм;
- робочий струм: 26 мА (5 В), 13 мА (3,3 В);
- кількість символів: 16 × 2;
- розмір символу: (2,96 × 5,56) мм;
- висота символів: (3,55 × 5,96) мм.

2.4 Датчик вологості та температури DHT11: його характеристики і особливості.

DHT11 [32] – це недорогий цифровий датчик для вимірювання температури та вологості. Датчик можна легко підключити до будь-якого мікроконтролера, або мінікомп’ютера, для миттєвого вимірювання вологості та температури.

Датчик вологості та температури DHT11 доступний у вигляді мікросхеми та модуля. Основна відмінність між мікросхемою та модулем полягає у наявності підтягуючого резистора та світлодіода живлення. DHT11 є датчиком відносної вологості, використовуючи термістор і ємнісний датчик вологості для вимірювання характеристик навколишнього повітря.



а) б)

Рис. 2.4.1 Датчик вологості та температури DHT11 у вигляді мікросхеми а) та модуля б)

Технічні характеристики **[32]*:***

- джерело живлення: (3,3 ÷ 5) В;
- струм споживання: макс. 2,5 мА;
- робочий діапазон: вологість та температура (20÷80) % RH, (0÷50) °C;
- діапазон вимірювання вологості: (20÷90) % RH;
- точність вимірювання вологості: ±5% RH;
- діапазон вимірювання температури: (0÷50) °C;
- точність вимірювання температури: ±2 °C;
- час відгуку: 1с;
- частота дискретизації: 1 Гц (1 вибірка в секунду);
- формат виведення даних: цифровий сигнал однієї шини;
- дальність передачі даних: (20÷30) м (на відкритому повітрі);
- розміри: (15 х 12 х 5,5) мм;
- вага: 2,5 г.

Функції пінів мікросхеми:

- Vcc – джерело живлення (3,5÷5,5) В;
- Data – виводить як температуру, так і вологість через послідовні дані;
- NC – не використовується;
- GND – земля.

Функції пінів модуля:

- Vcc – джерело живлення (3,5÷5,5) В;
- Data – виводить як температуру, так і вологість через послідовні дані;
- GND – земля.

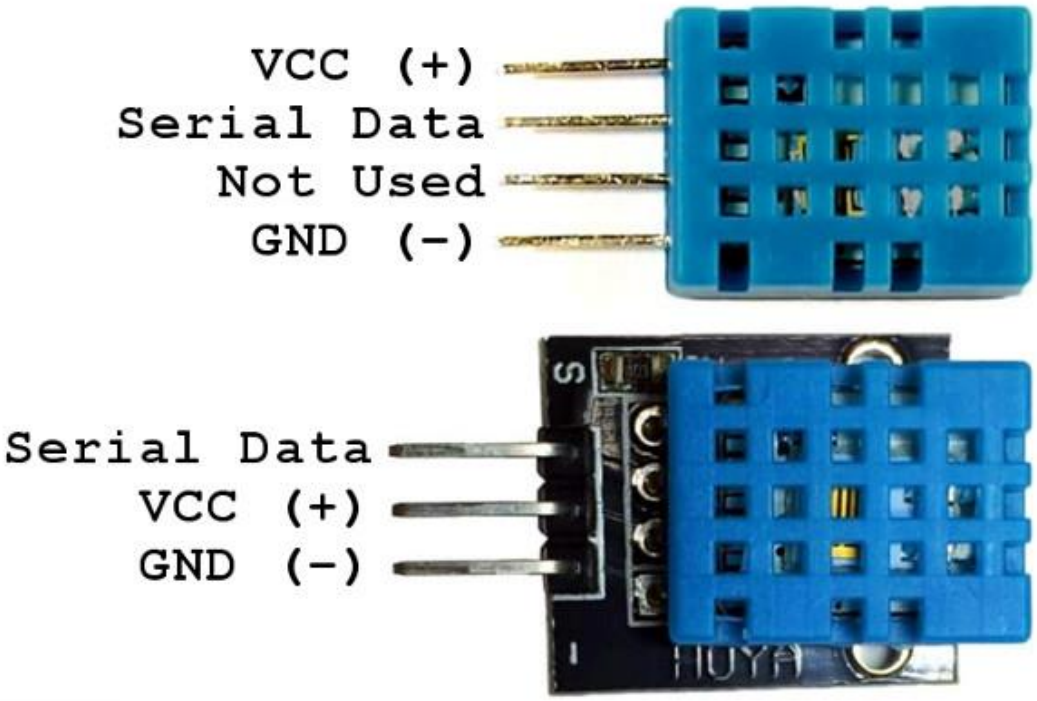


Рис. 2.4.2 Розпіновка датчика вологості та температури DHT11

Принцип роботи датчика DHT11.

Датчик DHT11 включає в себе елемент вимірювання вологості на основі ємності та термістор для вимірювання температури. У структурі датчика вологості конденсатор має два електроди з вологоутримуючою підкладкою як діелектриком між ними, і його ємність змінюється в залежності від рівня вологості. Вбудований інтегральний пристрій вимірює та обробляє змінені значення опору, перетворюючи їх у цифровий формат.

Для вимірювання температури, датчик використовує термістор із негативним температурним коефіцієнтом, що призводить до зменшення опору при збільшенні температури. Для забезпечення сталості значення опору при найменших змінах температури зазвичай використовують напівпровідникову кераміку або полімери.

Висновки

У даному розділі було розглянуто модулі, які використовувались у створені смарт-годинника, а саме: мікроконтролерна плата Arduino Uno, датчик вологості та температури DH11, годинник реального часу RTC DS1302, символьний рідкокристалічний дисплей LCD 1602 I2C.

Були представлені технічні характеристики та можливості платформи Arduino

Uno, що визначається як важливий інструмент для реалізації інноваційних технічних рішень. Детально розглянуто годинники реального часу (RTC) DS3231 і DS1302, їхні особливості та характеристики. Також ретельно розглядався символьний рідкокристалічний дисплей LCD 1602 I2C, його характеристики, що є важливим елементом для відображення інформації в реальному часі в різних пристроях. Окремо був вивчений датчик вологості та температури DHT11, його технічні характеристики та особливості. Цей датчик є важливим компонентом для вимірювання параметрів оточуючого середовища, що має значення в багатьох галузях, таких як сільське господарство, промисловість, медицина та інші.

РОЗДІЛ 3 АПАРАТНО-ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Апаратна реалізація смарт-годинника.

3.1.1 Загальна схема смарт-годинника. Його апаратні особливості.

Реалізація смарт-годинника передбачала використання керуючого пристрою у вигляді мікроконтролерної плати Arduino Uno. Вибір плати був зумовлений її дешевизною, доступністю та технічним характеристикам, які повністю задовільнили поставленні завдання. Для написання коду використовувалось середовище розробки Arduino IDE.

Годинник передбачає наступні функції:

- показувати поточний час та дату;
- зберігати поточну дату та час попри знеструмлення всієї конструкції;
- показувати рівень вологості повітря та її температуру;
- секундомір;
- таймер.

Для реалізації даного функціоналу, в конструкцію годинника було додано наступні модулі:

- годинник реального часу RTC DS1302;
- датчик вологості та температури DH11;
- рідкокристалічний дисплей LCD 1602 I2C;
- 6 кнопок;
- пасивний зумер.

Апаратна реалізація проекта представляє собою наступну електричну принципову схему:

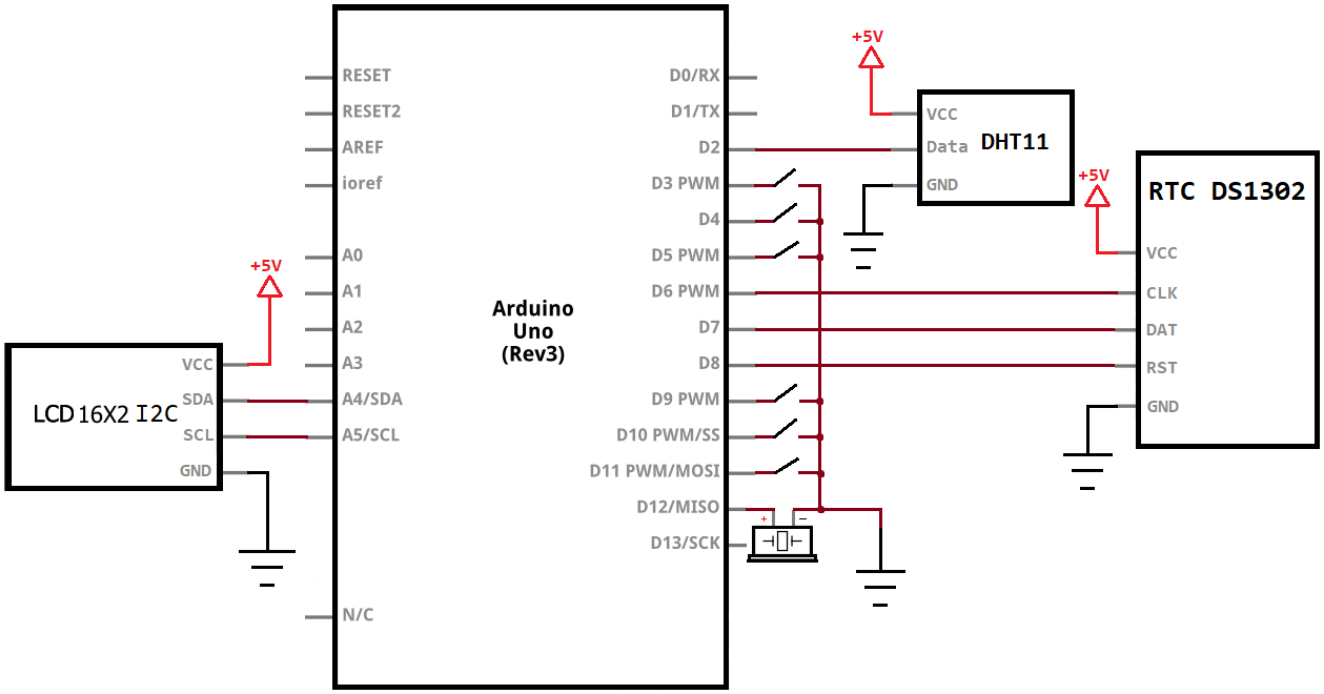


Рис. 3.1.1 Принципова схема смарт-годинника.

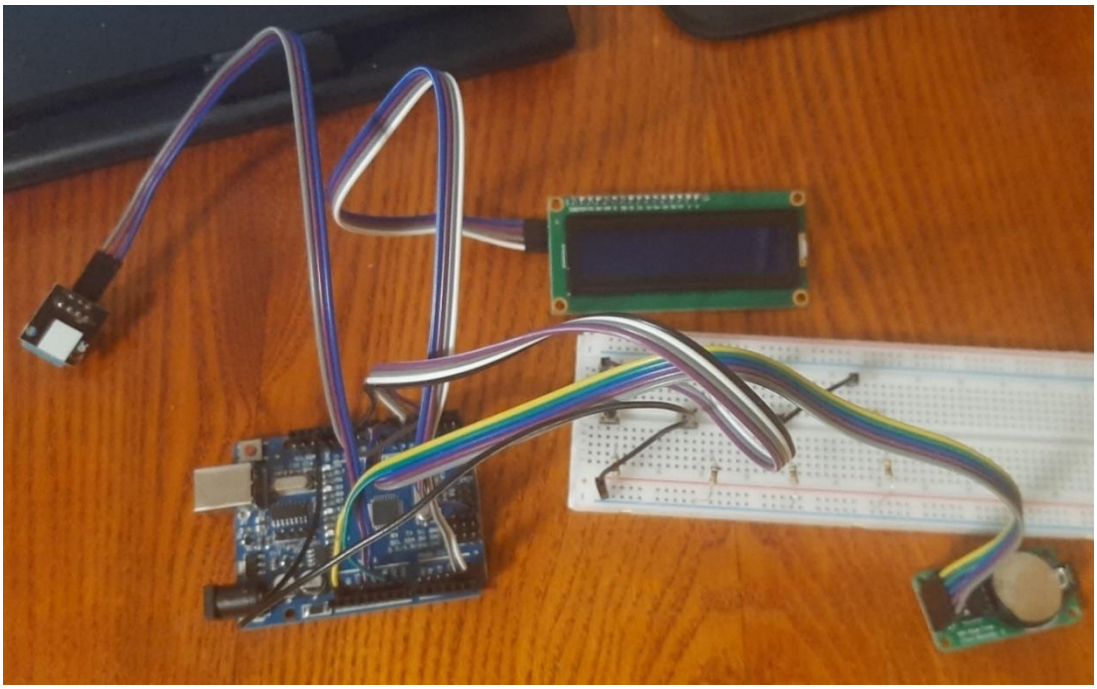


Рис. 3.1.2 Монтажна схема смарт-годинника.

У зібраному вигляді смарт-годинник має даний вигляд, де:

1. USB роз’єм, який використовується як один з варіантів подачі живлення для годинника і способу оновлення програмного забезпечення пристрою;
2. роз’єм для підключення від зовнішнього джерела живлення (7÷12)В;
3. пасивний зумер, який використовується для відтворення звуку при

натисканні на кнопку і для відтворення довгого звукового сигналу при завершенні роботи таймера;

4. кнопка включення/виключення;
5. 6 тактових кнопок, які використовуються для взаємодії з інтерфейсом керування приладом;
6. рідкокристалічний LCD дисплей, який використовується як засіб виведення інформації;
7. датчик температури та вологості.

Рис. 3.1.3 Смарт-годинник у зібраному вигляді.

3.1.2 Підключення символьного рідкокристалічного дисплея.

Як засіб виведення інформації годинник передбачає символьний рідкокристалічний дисплей. Для економії кількості задіяних пінів Arduino, використовувалася послідовна асиметрична шина.

Зважаючи на розпіновку дисплею, його підключення до Arduino UNO проводилась згідно наступної таблиці.

Табл. 3 Інтерфейс підключення Arduino UNO – LCD 1602 I2C.

Arduino	**LCD 1602 I2C**
GND	GND
5V	VCC
A4	SDA
A5	SCL

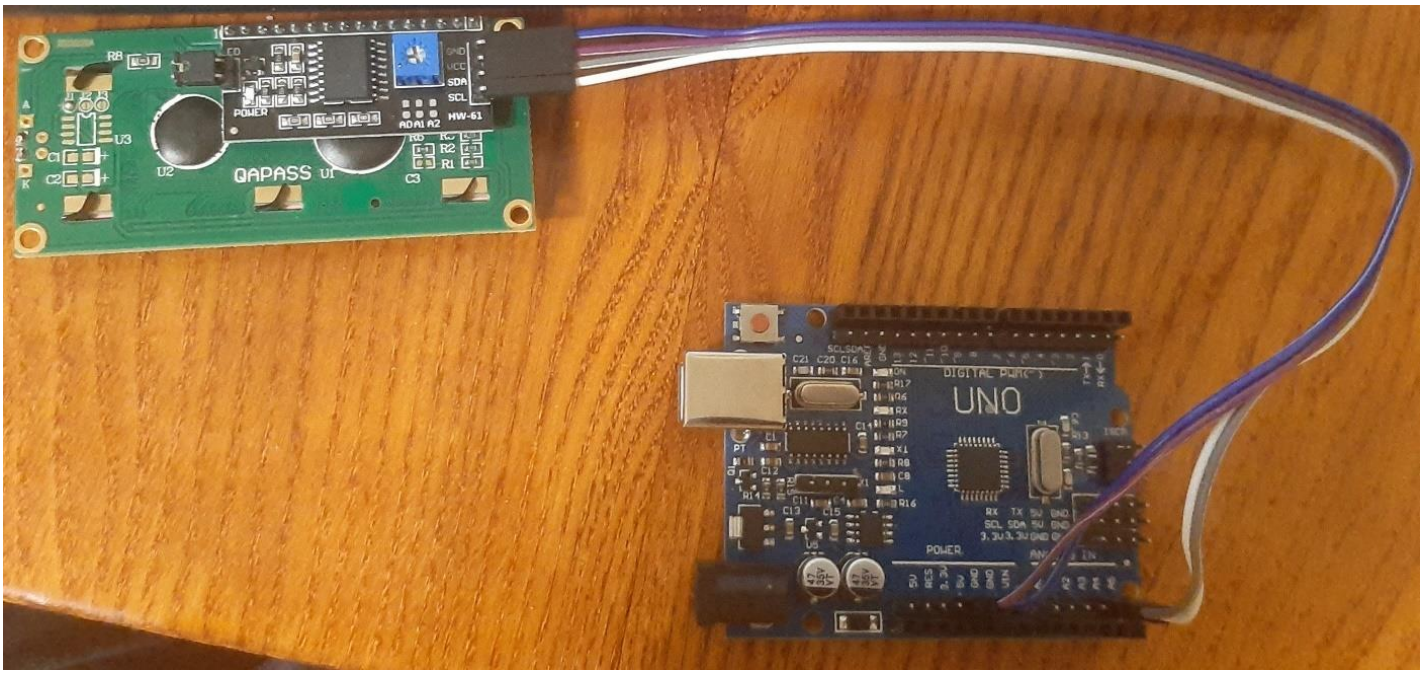


Рис. 3.1.4 Підключення модуля LCD 1602 I2C до Arduino UNO.

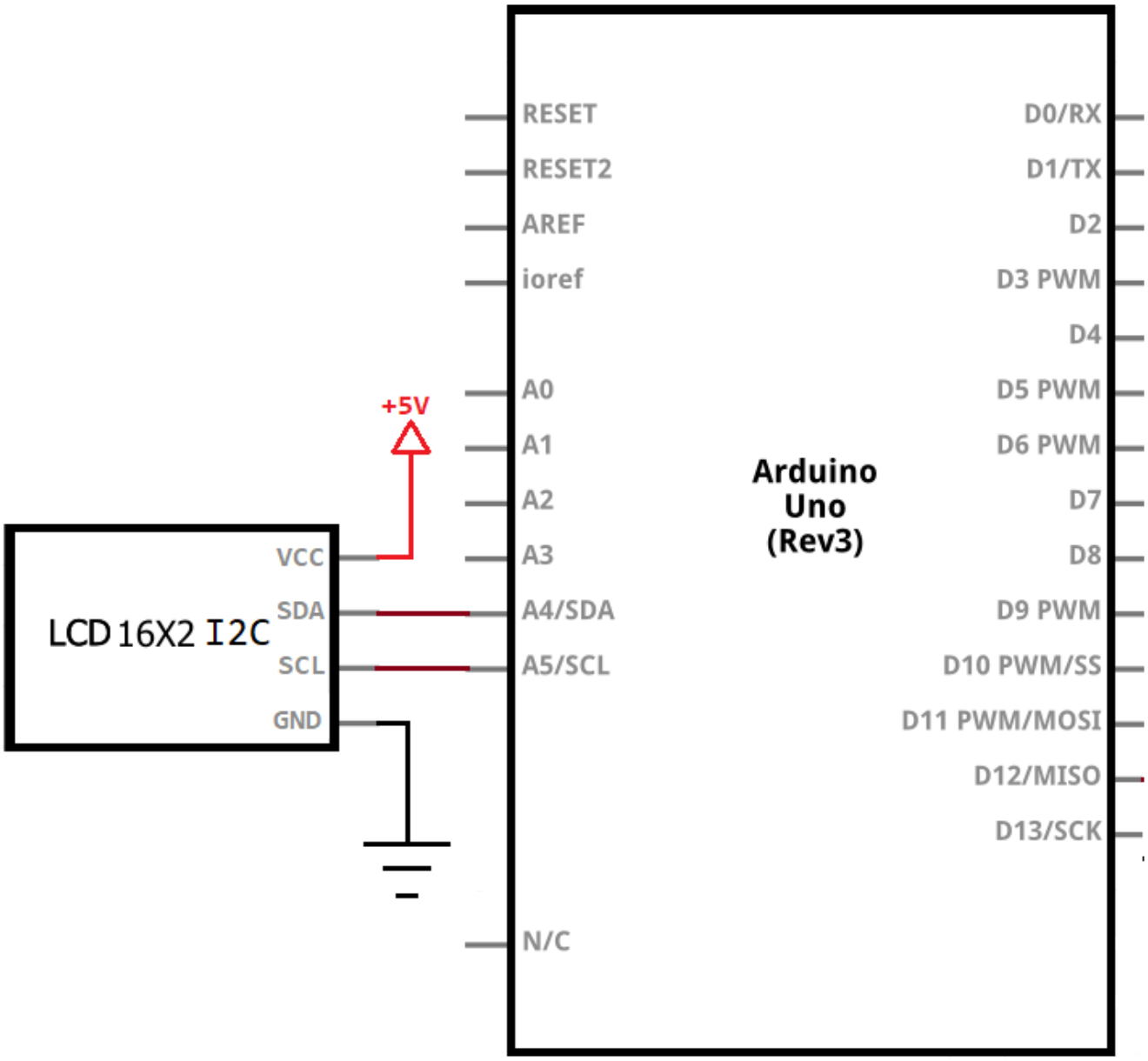


Рис. 3.1.5 Схема підключення модуля LCD 1602 I2C до Arduino UNO.

Далі проводилося програмне налаштування роботи Arduino з дисплеєм. Для цього використовувалися спеціалізовані бібліотеки *LiquidCrystal_I2C.h* і *Wire.h*. *LiquidCrystal_I2C.h* містить такі функції, як можливість очищення дисплею, переміщення курсору, створення символів, введення даних, тощо. В свою чергу *Wire.h*. дозволяє Arduino взаємодіяти з дисплеєм за інтерфейсом I2C/TWI.

Для налаштування яскравості дисплею необхідно скористатися регулюваним резистором, який розміщений на платі модуля дисплею.

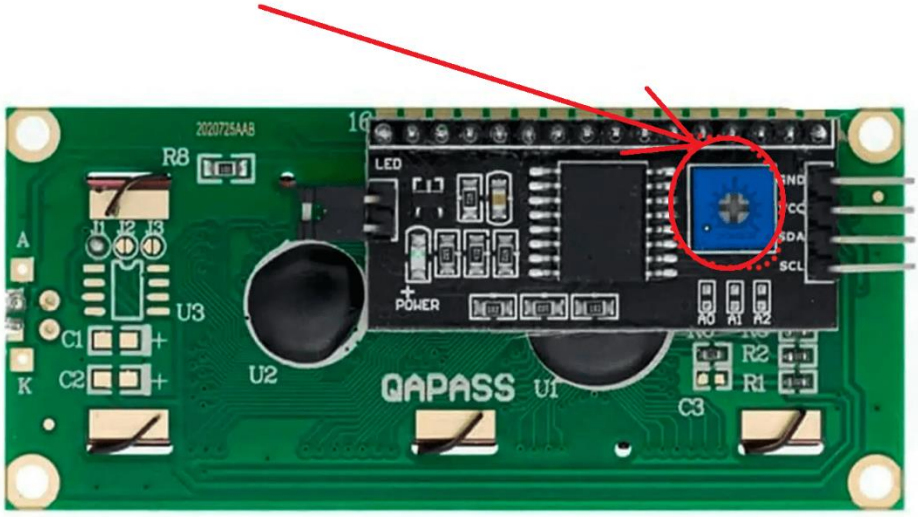


Рис. 3.1.6 Регулювальний резистор на модулі LCD 1602 I2C.

3.1.3 Підключення датчика вологості та температури.

Для збирання інформації про температуру та рівень вологості повітря використаний модуль DHT11.

Табл. 4 Інтерфейс підключення Arduino UNO – DHT11.

Arduino	**DHT11**
GND	GND
5V	VCC
D2	Data

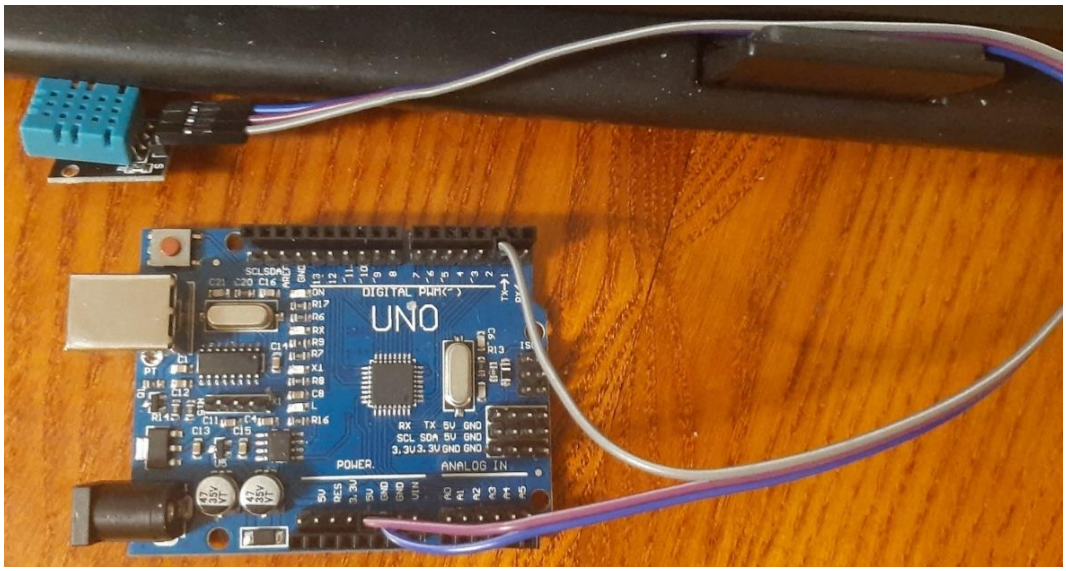


Рис. 3.1.7 Підключення модуля DHT11 до Arduino.

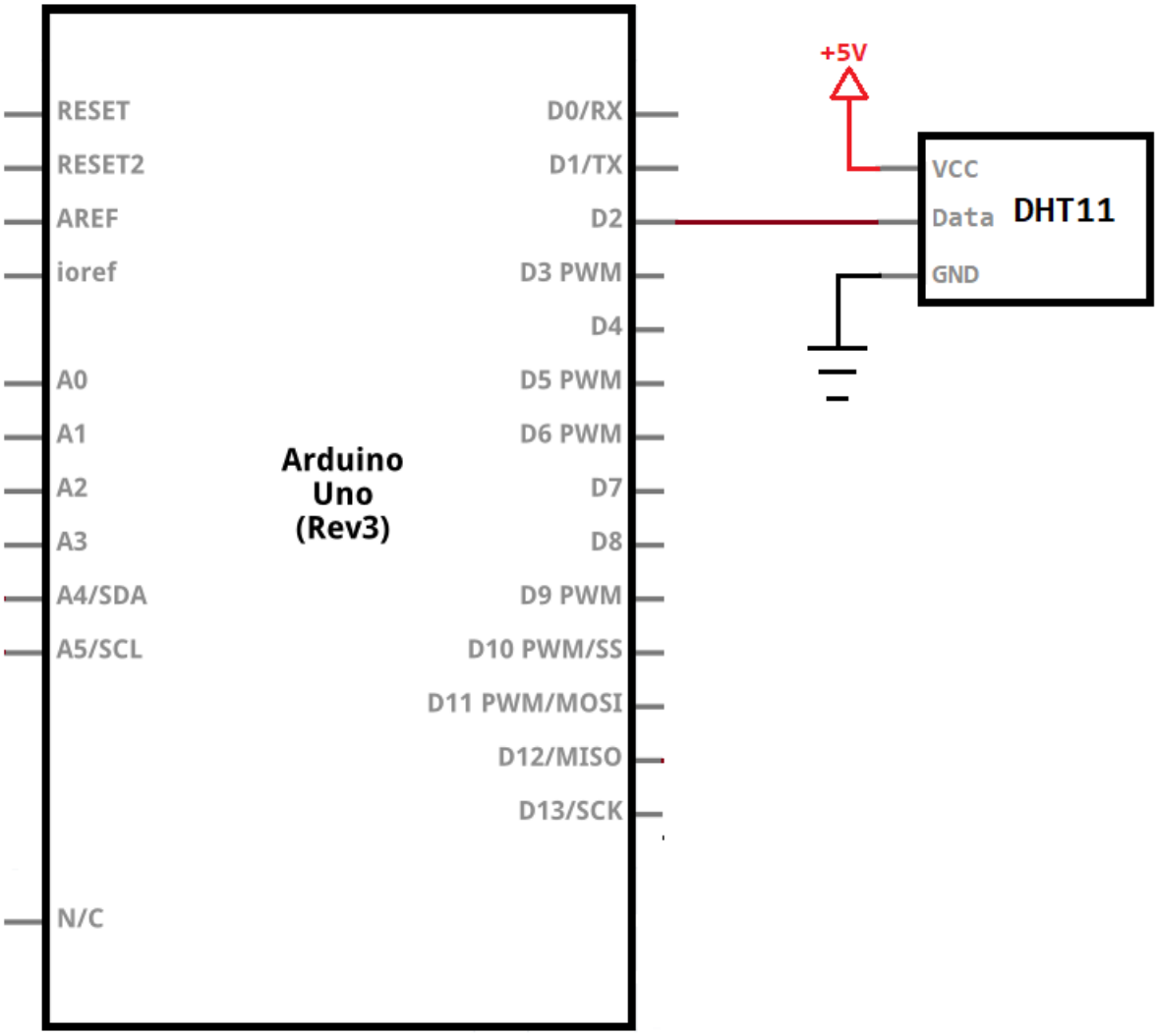


Рис. 3.1.8 Схема підключення модуля DHT11 до Arduino UNO.

Для роботи із модулем DHT11 використовувалася бібліотека *DHT.h*, яка дає можливість читати данні про температуру та вологість.

3.1.4 Підключення годинника реального часу.

В свою чергу, для зберігання даних про дату та час було використано годинник реального часу RTC DS1302. При цьому, підключення даного модуля до керуючої плати здійснювалося згідно наступної таблиці:

Табл. 5 Інтерфейс підключення Arduino UNO – RTC DS1302.

Arduino	**RTC DS1302**
GND	GND
5V	VCC

Продовження таблиці 5

D6	CLK
D7	DAT
D8	RST

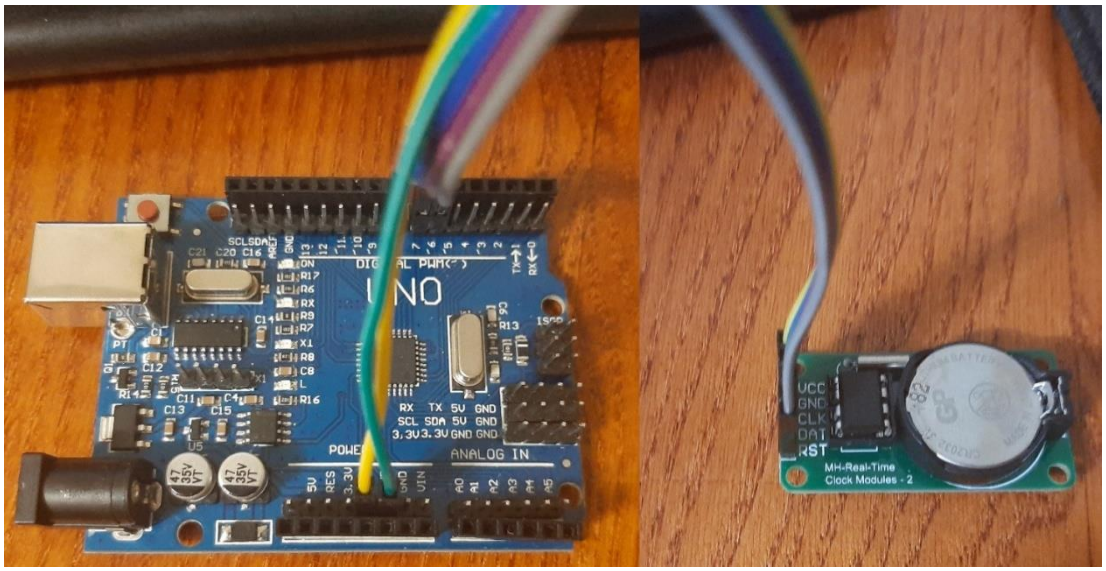


Рис. 3.1.9 Підключення модуля RTC DS1302 до Arduino UNO.

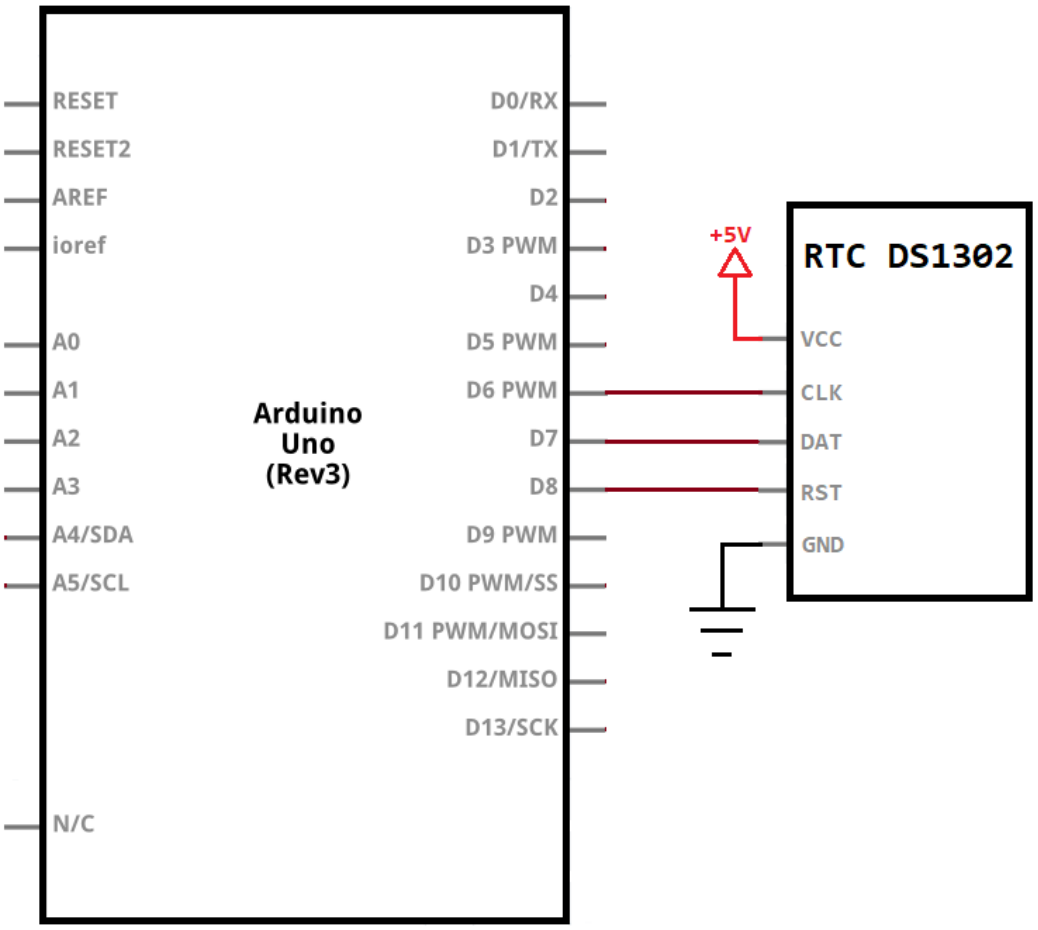


Рис. 3.1.10 Схема підключення модуля RTC DS1302 до Arduino UNO.

Для використання можливостей модуля RTC використовувались бібліотеки *ThreeWire.h* і *RtcDS1302.h*, функції яких дають можливість читати та редагувати дані про поточний час і дату.

3.1.5 Підключення тактових кнопок.

Інтерфейс керування приладом був реалізований за допомогою шести тактових кнопок, функціонал яких зав’язаний на переході між функціями годинника.

Тактові кнопки підключаються до Arduino безпосередньо до цифрових пінів: D3 – перша кнопка, D4 – друга кнопка, D5 – третя кнопка, D9 – четверта кнопка, D10 – п’ята кнопка, D11 – шоста кнопка. Всі інші кінці кнопок заземлюються.

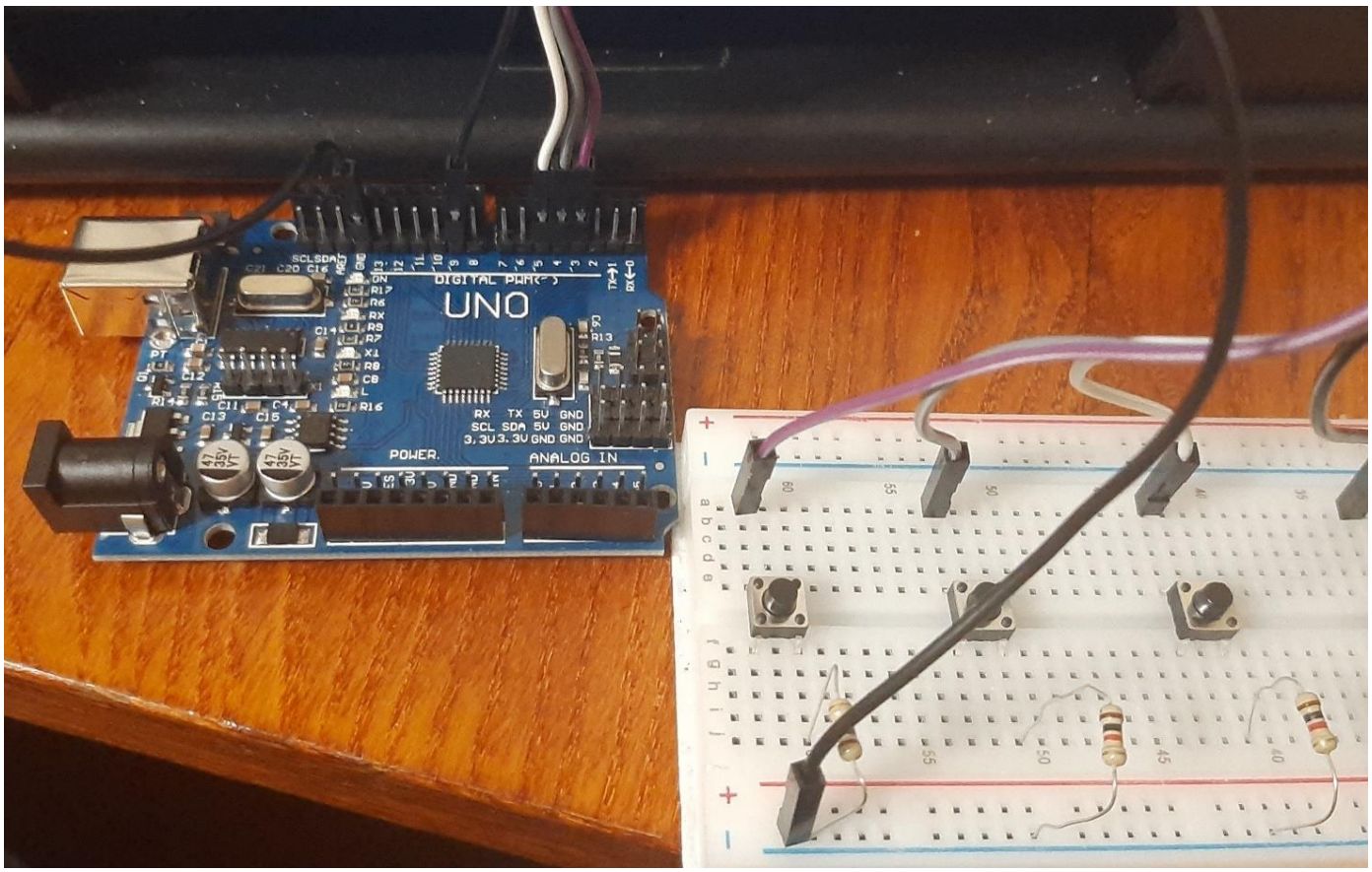


Рис. 3.1.11 Підключення тактових кнопок до Arduino UNO.

Така схема підключення, при натисненні/відтисненні на будь-яку із кнопок, дає можливість керуючому пристрою зчитати відповідний логічний рівень, цифрового порта.

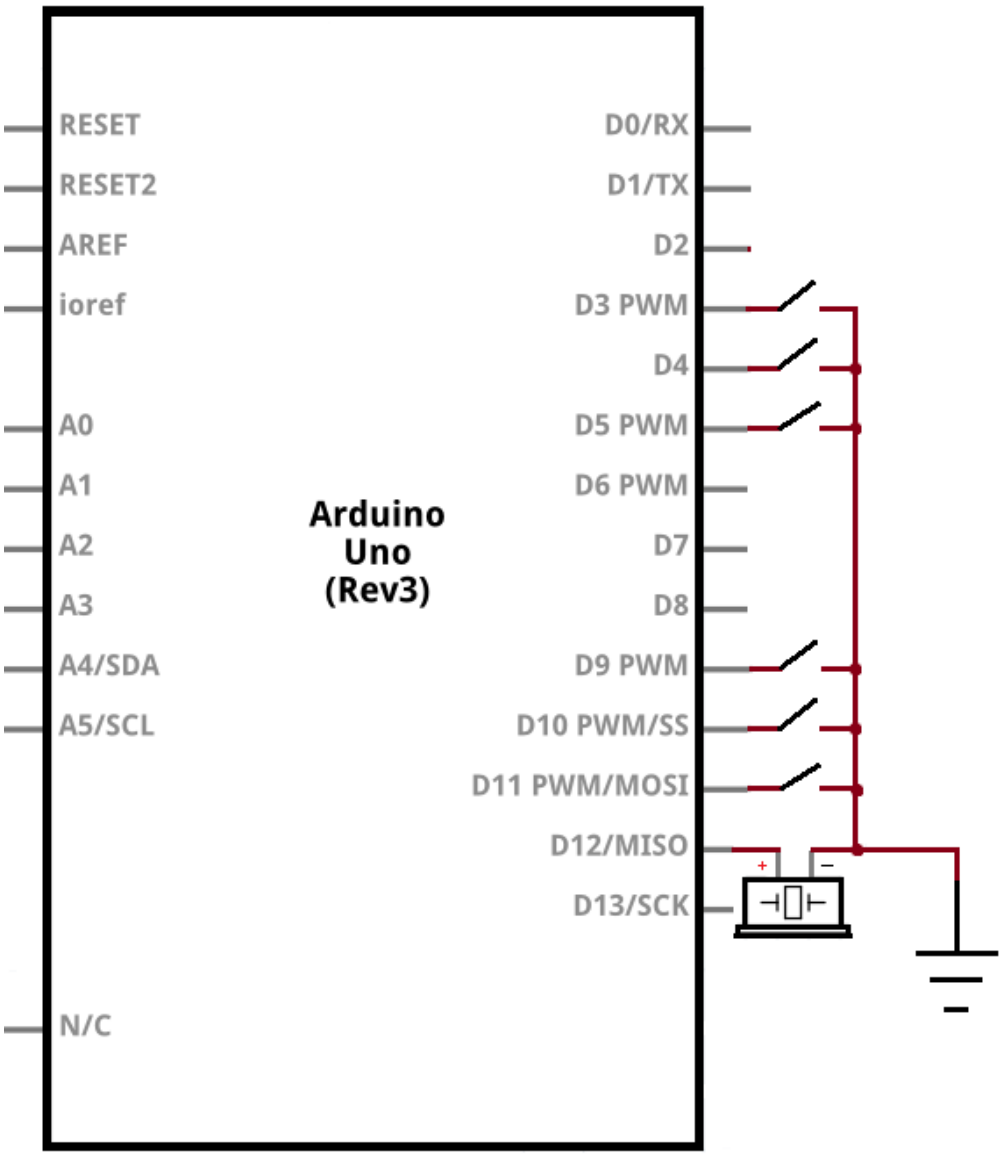


Рис. 3.1.12 Схема підключення тактових кнопок до Arduino UNO.

3.2 Програмна реалізація смарт-годинника.

3.2.1 Інструкція користування.

Першопочаткове увімкнення годинника передбачає засвічення екрану LCD дисплея голубим кольором. Для подальшої роботи необхідно вибрати одну із запрограмованих функцій цього годинника.

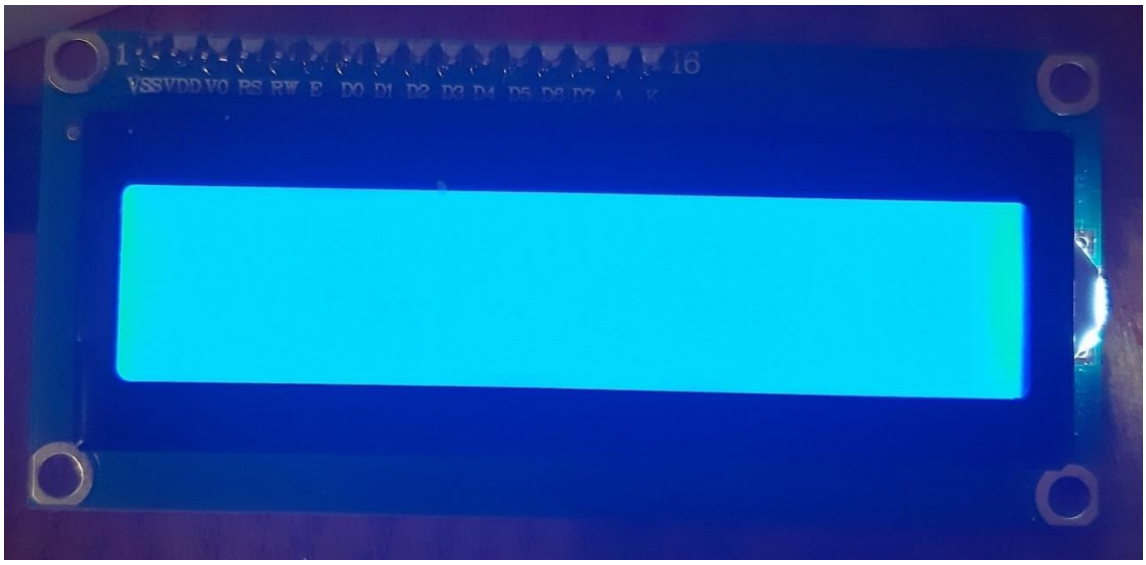


Рис. 3.2.1 LCD дисплей при включенні годинника.

Щоб скористатися однією з них, пропонується натискати одну з шести кнопок:

1. Секундомір.
2. Введення кількості годин для таймера.
3. Введення кількості хвилин для таймера.
4. Таймер
5. Огляд температури та вологості повітря.
6. Годинник.

При натисканні на шосту кнопку дисплей демонструє поточну дату та час, який оновлюється кожну секунду.

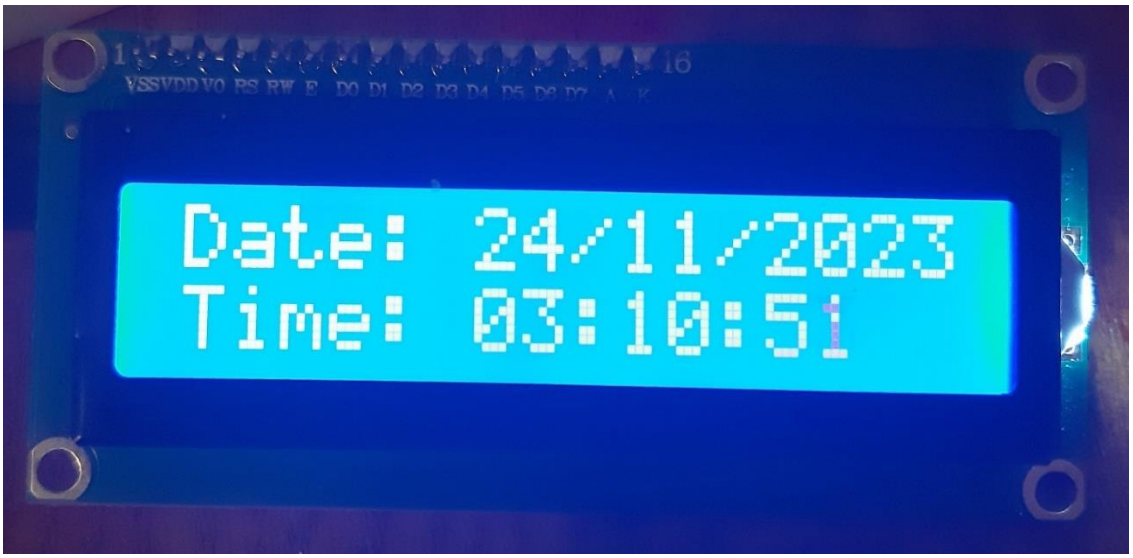


Рис. 3.2.2 LCD дисплей при натисканні шостої кнопки.

Виведених на дисплей дати і часу буде продовжуватися до моменту, доки не буде натиснута перша, друга, третя, четверта або п’ята кнопка.

При натискання на п’яту кнопку, дисплей демонструє температуру у градусах Цельсія та відносну вологість у відсотках.

Рис. 3.2.3 LCD дисплей при натисканні п’ятої кнопки.

При цьому інформація на екрані оновлюється кожні 2 секунди. Графічні дані на дисплеї будуть відтворюватися доти, поки не буде натиснута перша, друга, третя, четверта або шоста кнопка.

Натискання на другу кнопку демонструє введення часу для таймеру у годинах, а натискання на третю кнопку демонструє введення часу для таймеру у хвилинах.

Рис. 3.2.4 LCD дисплей при натисканні другої кнопки.

Рис. 3.2.5 LCD дисплей при натисканні третьої кнопки.

Натискання на четверту кнопку демонструє запущений таймер, який відразу запускає свій рахунок. Графічна інформація відображається у хвилинах, секундах та мілісекундах.

Рис. 3.2.6 LCD дисплей при натисканні четвертої кнопки.

З таймера вийти неможливо, поки він не буде поставлений на паузу чи не закінчиться час. Щоб поставити на паузу, достатньо ще раз натиснути на четверту кнопку. Тепер, якщо натиснути на першу, другу, третю, п’яту чи шосту кнопку, відбувається перехід на відповідній функціонал.

Натискання на першу кнопку демонструє запущений секундомір, який відразу запускає свій рахунок. Графічна інформація відображається у хвилинах, секундах та мілісекундах.

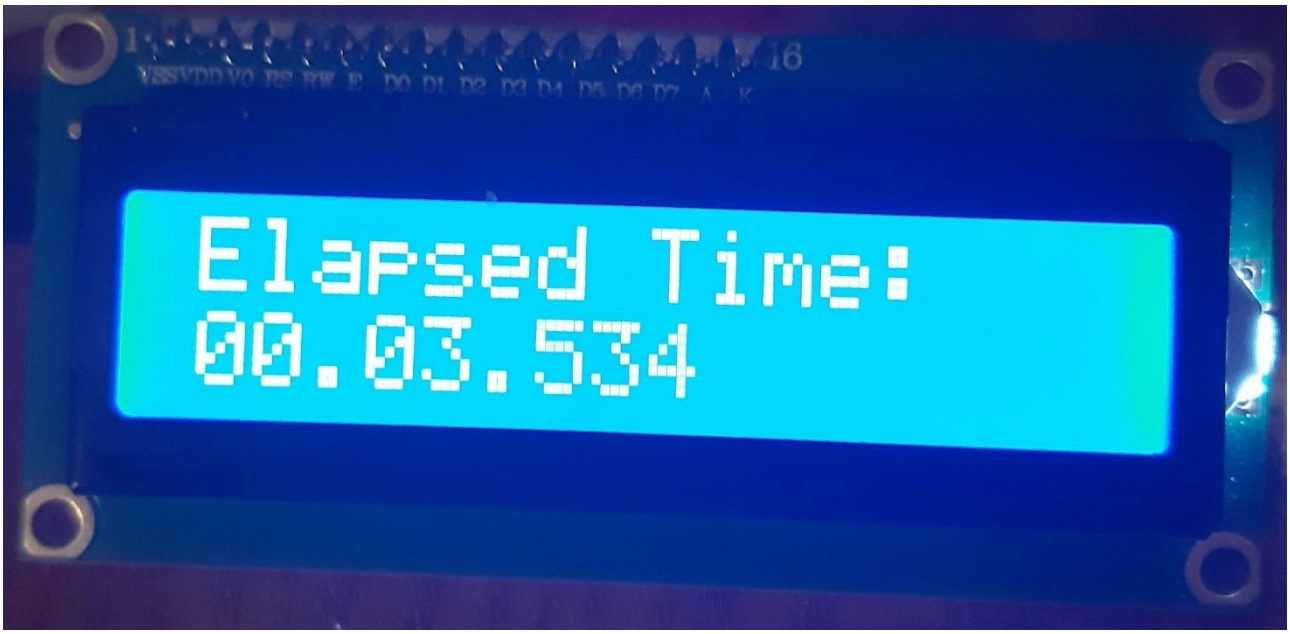


Рис. 3.2.7 LCD дисплей при натисканні першої кнопки.

З секундоміра вийти неможливо, поки він не буде поставлений на паузу. Щоб це зробити, достатньо ще раз натиснути на першу кнопку. Тепер, якщо натиснути на другу, третю, четверту, п’яту чи шосту кнопку, відбувається перехід на відповідній функціонал.

Якщо замість виходу з секундоміра натиснути ще раз на першу кнопку, секундомір обнулиться і запуститься ще раз. Слід зауважити, що послідовність переходу між режимами смарт-годинника є будь-якою.

3.2.2 Код смарт-годинника. Його пояснення.

Розглянемо більш детально код функціоналу 1 кнопки:

```
if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
  {
    tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
    test=1; /* Вказуємо змінній test значення 1. Це робиться для запуску циклу, який
буде відповідати за хід шкали таймера в реальному часі. */
    startTime = millis(); /* Ініціалізація таймера */

    while(test==1) /* Запускаємо цикл */
    {
      lcd.clear(); /* Чистимо інформацію на екрані дисплею */
      lcd.setCursor(0, 0); /* Вказуємо кординату дисплею для виведення інформації */

      unsigned long elapsedTime = millis() - startTime; /* Визначаємо час, який пройшов
від початку таймера */
      int minutes = (elapsedTime % 3600000) / 60000; /* Обчислення хвилин */
```

```
      int seconds = (elapsedTime % 60000) / 1000; /* Обчислення секунд */
      int milliseconds = elapsedTime % 1000; /* Обчислення мілісекунд */

      /* Форматуємо час для виведення на екран */
      String secondsString = String(seconds < 10 ? "0" + String(seconds) :
String(seconds)); /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо
на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення.
*/
      String minutesString = String(minutes < 10 ? "0" + String(minutes) :
String(minutes));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо
на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення.
*/
      String millisecondsString = String(milliseconds < 10 ? "0" + String(milliseconds)
: String(milliseconds));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми
виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це
значення. */

      lcd.print("Elapsed Time:"); /* виводимо даний текст на верхню частину дисплея.
Координати були вказані раніше. */
      lcd.setCursor(0, 1);  /* Вказуємо кординату дисплею для виведення інформації */
      lcd.print(String(minutesString) + "." + String(secondsString) + "." +
String(millisecondsString)); /* виводимо пройдений час у іншій половині дисплею. */
      delay(400); /* Оновлюємо інформацію кожні 400 мілісекунд. */

      if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
        {
          tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
          test=0; /* Присвоюємо інше значення. По цій причині секундомір зупиняється
(відбувається вихід з циклу). */
          delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд. */
        }
      }
    }
```

Тут на початку присвоюється стартове значення змінної *test*, від якої залежить наступний цикл роботи секундоміра, і ініціалізуємо таймер.

Далі запускається цикл, в якому задається змінна *elapsedTime*, яка дорівнює поточному часові, від якого відняли стартовий. Також вводяться *int* змінні, які відображають цей час у хвилинах, секундах та мілісекундах.

Далі вводяться *String* змінні, які відображають попередні *int* змінні тільки з урахуванням однієї умови, а саме, якщо це *int* змінна дорівнює менше 10, то перед кожним числом ставиться 0.

Далі за допомого *lcd.setCursor()* вказуються координати розміщення текста і виводу поточного часу секундоміра, а *lcd.print* виводить якраз дану інформацію на дисплей.

Команда *delay(400)* вказує на оновлення цієї інформації і її вивід кожні 400 мілісекунд.

Щоб вийти з цього циклу, здійснюється ще одна перевірка, а саме, якщо буде натиснута перша кнопка ще раз, то *test* приймає значення 0, що і виводить хід виконання програми з даного циклу.

Розглянемо більш детально код функціоналу 2 кнопки:

```
if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      selectedHour = (selectedHour + 1) % 24; /* Збільшення години на 1, обмеження до 23
(0-23 години). Значення збільшується при кожному натисканні. */
      lcd.clear(); /* Очищення дисплею */
      String HourString = String(selectedHour < 10 ? "0" + String(selectedHour) :
String(selectedHour));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми
виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це
значення. */
      lcd.setCursor(0, 0); /* Вказуємо координати */
      lcd.print("Hour: "); /* Виводимо текст */
      lcd.print(HourString); /* Виводимо значення вказаної раніше годинни */
      delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
    }
```

Тут на початку створюється змінна selectedHour, яка зберігає в собі значення години. При кожному повторному натисканні це значення зростає на 1. Далі проводиться перевірка виводу інформації, а саме чи це число менше 10, і якщо так, то на дисплей виводиться дане число з 0 поперед собою. Далі виводиться така інформація.

Розглянемо більш детально код функціоналу 3 кнопки:

```
if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      selectedMinute = (selectedMinute + 1) % 60; /* Збільшення хвилини на 1, обмеження
до 59 (0-59 хвилин) */
      lcd.clear(); /* Очищення дисплею */
      String MinuteString = String(selectedMinute < 10 ? "0" + String(selectedMinute) :
String(selectedMinute));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми
виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це
значення. */
      lcd.setCursor(0, 0);  /* Вказуємо координати */
      lcd.print("Minute: ");  /* Виводимо текст */
      lcd.print(MinuteString); /* Виводимо значення вказаної раніше хвилини */
      delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
    }
```

Тут все відбувається за схожим принципом, як при натисканні другої кнопки. Єдина різниця полягає в тому, що саме значення не в годинах, а у хвилинах.

Розглянемо більш детально код функціоналу 4 кнопки:

```
allMili = (selectedHour * 3600) + (selectedMinute * 60); /* Обчислення часу в секундах
*/
    lt = 1; /* Позначення, що таймер встановлено */

    if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четвертої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      while(lt == 1) /* Запускаємо цикл */
      {
        lcd.clear(); /* Очищення дисплею */
        int elapsedTime1 = allMili--; /* Зменшення часу */
        int hours = (elapsedTime1 % 86400) / 3600; /* Обчислення годин */
        int minutes = (elapsedTime1 % 3600) / 60; /* Обчислення хвилин */
        int seconds = (elapsedTime1 % 60); /* Обчислення секунд */

        String hoursString1 = String(hours < 10 ? "0" + String(hours) :
String(hours));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на
дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
        String secondsString1 = String(seconds < 10 ? "0" + String(seconds) :
String(seconds));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо
на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення.
*/
        String minutesString1 = String(minutes < 10 ? "0" + String(minutes) :
String(minutes));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо
на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення.
*/

        lcd.setCursor(0, 0);  /* Вказуємо координати */
        lcd.print("Timer Time:"); /* Виводимо текст */
        lcd.setCursor(0, 1);  /* Вказуємо координати */
        lcd.print(String(hoursString1) + "." + String(minutesString1) + "." +
String(secondsString1));  /* виводимо пройдений час таймеру у іншій половині дисплею. */
        delay(1000);  /* Оновлюємо інформацію кожну секунду */
        if (elapsedTime1 == 0) /* Перевірка на закінчення часу */
        {
          /* Генерація сигналу про закінчення таймера */
          tone(soundPin, 2000, 500); /* Відтворення звукового сигналу */
          delay(1000);  /* пауза на секунду */
          tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */
          delay(1000);  /* пауза на секунду */
          tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */
          delay(1000);  /* пауза на секунду */
          tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */

          lt = 0; /* Зупинка таймера */
          /* Обнулення всіх раніше введених значень. */
          elapsedTime1 = 0;
          selectedHour = 0;
          selectedMinute = 0;
          hours = 0;
          minutes = 0;
```

```
          seconds = 0;
          delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
        }
```

На початку вводиться змінна, яка зберігає в собі загальне значення всіх введених хвилин та годин. Також вводиться змінна для старту циклу.

Після натискання четвертої кнопки програється звук і запускається цикл. Далі із загального часового значення allMili дістається значення годин, хвилин та секунд.

Після цього, робиться перевірка виводу інформації, а саме чи ці числа менше 10, і якщо так, то на дисплей виводиться ці числа з 0 поперед собою. Далі виводиться таймер, циферблат якого оновлюється щосекунди. Вихід з циклу відбувається при 2-ох умовах: якщо таймер закінчив свій відлік і якщо була натиснута пауза. Щоб призупинити відлік таймеру достатньо натиснути повторно на четверту кнопку. Після цього, можна переходити на будь-який інший функціонал.

Розглянемо більш детально код функціоналу 5 кнопки:

```
if (digitalRead(switchPin5) == LOW) /* Перевірка на натискання п'ятої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      lcd.clear(); /* Очищення дисплею */
      ns = 1; /* Вказуємо змінній ns значення 1. Це робиться для запуску циклу. */
      while(ns == 1) /* Запускаємо цикл */
      {
        if (isnan(h) || isnan(t))  /* Перевірка на помилку */
        {
          lcd.print("err"); /* вивід тексту */
          delay (3000); /* оновлювати кожні 3 секунди */
          lcd.clear(); /* Очищення дисплею */
          return;
        }
        else
        {
          lcd.setCursor(0, 0); /* вказуємо координати */
          lcd.print("temp: "); /* вивід тексту */
          lcd.print(t); /* вивід температури */
          lcd.print(char(223)); /* вивід знака градусів */
          lcd.print("C"); /* вивід тексту */
          lcd.setCursor(0, 1); /* вказуємо координати */
          lcd.print("hum: "); /* вивід тексту */
          lcd.print(h); /* вивід вологості */
          lcd.print(char(37)); /* вивід знака відсотків */
          delay (2000); /* оновлювати давнні кожні 2 секунди */
          lcd.clear(); /* вичищення дисплею */
          if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки
*/
          {
            ns = 0; /* вихід з циклу */
```

```
          }
          if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки
*/
          {
            ns = 0; /* вихід з циклу */
          }
          if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки
*/
          {
            ns = 0; /* вихід з циклу */
          }
          if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четверту кнопки
*/
          {
            ns = 0; /* вихід з циклу */
          }
          if (digitalRead(switchPin6) == LOW) /* Перевірка на натискання шостої кнопки
*/
          {
            ns = 0; /* вихід з циклу */
          }
        }
      }
    }
```

Тут за схожим принципом вводиться на початку змінна *ns*, яка дорівнює 1, а потім запускається цикл, всередині якого проводиться перевірка, чи значення *h* і *t* (вологість і температура, відповідно) дорівнюють 0. Якщо так, то виводиться інформація про помилку. В іншому випадку виводиться інформація про температуру і вологість на даний момент. Така інформація оновлюється кожні 2 секунди. Сам цикл вводився для того, щоб реалізувати моментальний перехід між функціями. Вихід з циклу відбувається за схожим принципом. Різниця лише полягає у тому, що задля виходу з нього необхідно натиснути першу, другу, третю, четверту або шосту кнопку.

Розглянемо більш детально код функціоналу 6 кнопки:

```
if (digitalRead(switchPin6) == LOW) /* Перевірка на натискання шостої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      lcd.clear(); /* Очищення дисплею */
    var = 1; /* значення для старту циклу */
    while(var == 1) /* запуск циклу */
    {
      RtcDateTime now = Rtc.GetDateTime();  /* Отримання поточної дати та часу з модуля
RTC */
      int hour = now.Hour(); /* введеня години */
      int minute = now.Minute(); /* введення хвилини */
      int second = now.Second(); /* введеня секунди */
```

```
      int year = now.Year(); /* введеня року */
      int month = now.Month(); /* введення місяця */
      int day = now.Day(); /* введення дня */

      String hourString = String(hour < 10 ? "0" + String(hour) : String(hour));/*
Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з
0 поперед собою. В іншому випадку просто виводиться це значення. */
      String minuteString = String(minute < 10 ? "0" + String(minute) :
String(minute));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на
дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String secondString = String(second < 10 ? "0" + String(second) :
String(second));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на
дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String monthString = String(month < 10 ? "0" + String(month) : String(month));/*
Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з
0 поперед собою. В іншому випадку просто виводиться це значення. */
      String dayString = String(day < 10 ? "0" + String(day) : String(day));/* Перевірка
на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед
собою. В іншому випадку просто виводиться це значення. */

      String dateString = String(dayString) + "/" + String(monthString) + "/" +
String(year); /* розтановка значень для ввиводу */
      String timeString = String(hourString) + ":" + String(minuteString) + ":" +
String(secondString); /* розтановка значень для ввиводу */
      lcd.setCursor(0, 0); /* вибід координати */
      lcd.print("Date: "); /* вивід тексту */
      lcd.print(dateString); /* вивід дати */
      lcd.setCursor(0, 1); /* вибір координати */
      lcd.print("Time: "); /* вивід тексту */
      lcd.print(timeString); /* вивід часу */
      delay(1000); /* оновлювати щосекундно */
      lcd.clear(); /* Очищення дисплею */
      if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четвертої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin5) == LOW) /* Перевірка на натискання п'ятої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
    }
```

```
    }
}
```

Код функції годинника дуже схожий на 2 попередніх. Так само вводиться змінна, єдине унікальне на початку, це Rtc.GetDateTime(), що бере поточну дату та час з модуля RTC. Далі так само вводяться змінні, які підтягують інформацію за рік, місяць, день, годину, хвилину, секунду. Та сама перевірка змінних на значення нижче 10, також вивід цієї інформації, який обновляється кожну секунду. І також той же самий вихід з циклу, тільки для цього необхідно натиснути першу, другу, третю, четверту або п’яту кнопку.

Висновки

В даному розділі досліджено апаратно-програмну реалізацію смарт-годинника. Починаючи з загальної схеми пристрою та його апаратних особливостей, вивчено підключення ключових компонентів, таких як символьний рідкокристалічний дисплей, датчик вологості та температури, годинник реального часу та тактові кнопки.

У розділі також розглянуто програмну реалізацію смарт-годинника, включаючи інструкцію користування та код пристрою з його поясненням. Детально висвітлено функціональні можливості смарт-годинника та способи їх використання.

Результати дослідження вказують на успішну інтеграцію апаратних компонентів з програмним забезпеченням, що дозволяє смарт-годиннику ефективно виконувати свої функції. Розглянуті аспекти апаратно-програмної реалізації надають зрозумілу базу для подальших досліджень та вдосконалення смарт-годинника в майбутньому.

ЗАГАЛЬНІ ВИСНОВКИ

В ході магістерської роботи було досліджено особливості створення смарт-годинника на основі мікроконтролерної плати Arduino Uno. Розглянуто переваги і недоліки даної реалізації.

Робота поділена на три розділи. У першому розділі було розглянуто історію створення та розвитку смарт годинників, їх ключові особливості та функції.

Другий розділ присвячений об’єктам і методам досліджень. Було детально розглянуто плату управління Arduino Uno, а також модулі, які використовувались у створенні смарт-годинника, а саме датчик температури та вологості DHT11, годинник реального часу RTC DS1302, символьний рідкокристалічний дисплей LCD 1602 I2C. Розглянуто їх характеристики, будову та особливості.

У третьому розділі описана апаратна реалізація смарт-годинника, зокрема детальна схема підключення всіх модулів. Представлена інструкція користувача. Також було детально розглянуто код реалізації функціоналу годинника.

Додаток містить програмну реалізацію проекту, яка забезпечує функціонал смарт-годинника. Код містить детальний коментар задля можливості майбутньої модифікації алгоритму роботи годинника.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ben Lutkevich, Alyssa Provazza. Definition smartwatch. *TeachTarget.* URL: https://www.techtarget.com/iotagenda/definition/smartwatch (дата звернення: 09.10.2023).
2. Wallace Jackson. SmartWatch Design Fundamentals: WatchFace Design for Samsung Galaxy SmartWatches. Apress, 2019. 446 p.
3. Функції та налаштування розумного годинника. *Gelius.* URL: https://gelius.ua/blog/kak-rabotaut-smart-chasy/ (дата звернення: 09.10.2023).
4. Bertrand Hochet, Antonio J. Acosta, Manuel J. Bellido. Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation. Springer Science & Business Media, 2002. 496 p.
5. Christian Piguet. Low-power Electronics Design. CRC, 2004. 912 p.
6. Blake Buettner. From Deep in the Watch Box: The Seiko Data 2000. *Worn and Wound*. URL: https://wornandwound.com/from-deep-in-the-watch-box-the-seiko-data-2000/ (дата звернення: 11.10.2023).
7. Vilius Petkauskas. Before smartwatches broke mainstream: list of vintage wrist computers. *Cybernews*. URL: https://cybernews.com/editorial/before-smartwatches-broke-mainstream-list-of-vintage-wrist-computers/ (дата звернення: 11.10.2023).
8. Cole Pennington. Just Because The Forgotten Glory Of The Timex Datalink. *Hodinkee*. URL: https://www.hodinkee.com/articles/the-forgotten-glory-of-the-timex-datalink (дата звернення: 12.10.2023).
9. Was the Seiko Ruputer the World’s First Smartwatch? *I know watches.* URL: https://iknowwatches.com/seiko-ruputer-world-first-smartwatch/ (дата звернення: 13.10.2023).
10. Nick T. Did you know that Samsung announced a watch phone in 1999? *Phone Arena*. URL: https://www.phonearena.com/news/Did-you-know-that-Samsung-announced-a-watch-phone-in-1999_id69376 (дата звернення: 13.10.2023).
11. WatchPad 1.5. *Way Back Machine*. URL: https://web.archive.org/web/20011205071448/http://www.trl.ibm.com/projects/ngm/index_e.htm (дата звернення: 13.10.2023).

12. John Biggs. Review – Fossil Wrist PDA. *Wrist Watch Review*. URL: https://www.wristwatchreview.com/review-fossil-wrist-pda/ (дата звернення: 14.10.2023).
13. Fossil Wrist PDA Watch. *Phonedb*. URL: https://phonedb.net/index.php?m=device&id=8477&c=fossil_wrist_pda_watch (дата звернення: 14.10.2023).
14. Jon Mentor. 2004 Microsoft SPOT watch smartwatch review. *Wear*. URL: https://wear.guide/smartwatches/2004-microsoft-spot-watch-smartwatch/ (дата звернення: 14.10.2023).
15. Ambrose Karella. Microsoft SPOT and the Beginning of Smartwatches. *Medium*. URL: https://medium.com/digitalshroud/microsoft-spot-and-the-beginning-of-smartwatches-e29be21f0d33 (дата звернення: 14.10.2023).
16. Jeremy Roche. Sony Ericsson Bluetooth Watch MBW-100 review: Sony Ericsson Bluetooth Watch MBW-100. *Cnet*. URL: https://www.cnet.com/reviews/sony-ericsson-bluetooth-watch-mbw-100-review/ (дата звернення: 15.10.2023).
17. Conrad Quilty-Harper. Sony Ericsson's MBW-100 Bluetooth watch reviewed. *Engadget.* URL: https://www.engadget.com/2006-10-28-sony-ericssons-mbw-100-bluetooth-watch-reviewed.html (дата звернення: 15.10.2023).
18. Anand Lal Shimpi. Samsung's S9110 Watchphone: What Came Before the Galaxy Gear. *AnandTech*. URL: https://www.anandtech.com/show/7278/samsungs-s9110-watchphone-what-came-before-the-galaxy-gear (дата звернення: 16.10.2023).
19. Richard Lai. Sony Ericsson LiveView review. *Engadget.* URL: https://www.engadget.com/2010-12-01-sony-ericsson-liveview-review.html (дата звернення: 16.10.2023).
20. Dan Bowman. Vyzin Electronics Private Limited Launches "ZigBee Remote Health Monitoring Watch with Personal Emergency Response System. *Fierce healthcare*. URL: https://www.fiercehealthcare.com/healthcare/vyzin-electronics-private-limited-launches-zigbee-remote-health-monitoring-watch (дата звернення: 17.10.2023).

21. Devindra Hardawar. Pebble smartwatch now shipping, but expect to wait a while for yours. *VentureBeat*. URL: https://venturebeat.com/business/pebble-smartwatch-shipping/ (дата звернення: 17.10.2023).
22. Chris Burns. Omate TrueSmart smartwatch hands-on: SIM-toting shooter in the wild. *Slash Gear*. URL: https://www.slashgear.com/omate-truesmart-smartwatch-hands-on-sim-toting-shooter-in-the-wild-07312054 (дата звернення: 17.10.2023).
23. Alex Colon. Samsung Gear 2 Review. PCmag. URL: https://www.pcmag.com/reviews/samsung-gear-2 (дата звернення: 18.10.2023).
24. Rene Ritchie. Apple Watch 2015 review. *iMore*. URL: https://www.imore.com/apple-watch-0-review (дата звернення: 18.10.2023).
25. Not a Smartwatch but a Watch That's Smart: Razer Announces Feature-Packed Digital Watch with Smart Functions. *Razer*. URL: https://press.razer.com/product-news/not-a-smartwatch-but-a-watch-thats-smart-razer-announces-feature-packed-digital-watch-with-smart-functions/ (дата звернення: 19.10.2023).
26. Massimo Banzi, Michael Shiloh. Getting Started with Arduino: The Open Source Electronics Prototyping Platform. Make Community, LLC, 2015. 245 p.
27. Jeremy Blum. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley, 2013. 384 p.
28. Michael Margolis. Arduino Cookbook. O'Reilly Media, Incorporated, 2012. 699 p.
29. Arduino®. Product Reference Manual SKU: A000066. 20.12.2023. Datasheet.
30. Maxim integrated. DS1302 Trickle-Charge Timekeeping Chip. 2015. Datasheet.
31. PSoC® Creator™. Character LCD with I2C Interface (I2C LCD) 1.10. 2016. Datasheet.
32. D-Robotics UK. DHT11 Temperature & Humidity Sensor. 30.07.2023. Datasheet.

ДОДАТОК

Код годинника:

```
#include <DHT.h> /* Підключення бібліотеки для роботи з датчиком вологості і температури DHT */
#define DHTPIN 2 /* Визначення піна, на якому підключено датчик DHT */
#include <Wire.h> /* Підключення бібліотеки для роботи з I2C протоколом */
#include <LiquidCrystal_I2C.h>  /* Підключення бібліотеки для роботи з I2C LCD дисплеєм */
#include <ThreeWire.h>  /* Підключення бібліотеки для використання трьохпровідкового з'єднання до модуля RTC DS1302 */
#include <RtcDS1302.h>  /* Підключення бібліотеки для роботи з модулем реального часу DS1302 */

LiquidCrystal_I2C lcd(0x27, 16, 2);  /* Ініціалізація об'єкту для роботи з LCD дисплеєм */
DHT dht(DHTPIN, DHT11);  /* Ініціалізація об'єкту для роботи з датчиком DHT11 */
ThreeWire myWire(7, 6, 8);  /* Ініціалізація об'єкту для підключення пінів DAT, CLK, RST годинника реального часу DS1302 */
RtcDS1302<ThreeWire> Rtc(myWire);  /* Ініціалізація об'єкту для роботи з модулем реального часу */

int var, test, ns, lt, lt1;  /* Змінні для керування функціоналом */
int soundPin = 12;  /* Пін для виведення звукового сигналу */
int switchPin1 = 3;  /* Пін для першої кнопки */
int switchPin2 = 4;  /* Пін для другої кнопки */
int switchPin3 = 5;  /* Пін для третьої кнопки */
int switchPin4 = 9;  /* Пін для четвертої кнопки */
int switchPin5 = 10;  /* Пін для п'ятої кнопки */
int switchPin6 = 11;  /* Пін для шостої кнопки */
unsigned long startTime = 0;  /* Початковий час для вимірювання інтервалу */
unsigned long allMili = 0;  /* Початковий час для зворотного відліку */
int selectedHour = 0;  /* Змінна для зберігання вибраних годин */
int selectedMinute = 0;  /* Змінна для зберігання вибраних хвилин */
int selectedSecond = 0;  /* Змінна для зберігання вибраних секунд */

void setup()
{
  Serial.begin(9600);  /* Ініціалізація зв'язку зі зовнішнім пристроєм ( комп'ютером) */
  dht.begin();  /* Ініціалізація датчика DHT */
  lcd.begin();  /* Ініціалізація LCD дисплею */
  delay(1000);  /* Затримка для стабілізації */
  lcd.backlight();  /* Включення підсвічування LCD дисплея */
  Rtc.Begin();  /* Ініціалізація модуля реального часу */
  Rtc.SetIsRunning(true);  /* Запуск модуля реального часу */
  pinMode(switchPin1, INPUT);  /* Встановлення піна для першої кнопки як вхід */
  digitalWrite(switchPin1, HIGH);  /* Підтягування до HIGH для уникнення неправильних сигналів */
  pinMode(switchPin2, INPUT);  /* Встановлення піна для другої кнопки як вхід */
  digitalWrite(switchPin2, HIGH); /* Підтягування до HIGH для уникнення неправильних сигналів */
  pinMode(switchPin3, INPUT); /* Встановлення піна для третьої кнопки як вхід */
  digitalWrite(switchPin3, HIGH); /* Підтягування до HIGH для уникнення неправильних сигналів */
  pinMode(switchPin4, INPUT); /* Встановлення піна для четвертої кнопки як вхід */
  digitalWrite(switchPin4, HIGH); /* Підтягування до HIGH для уникнення неправильних сигналів */
```

```
  pinMode(switchPin5, INPUT); /* Встановлення піна для п'ятої кнопки як вхід */
  digitalWrite(switchPin5, HIGH); /* Підтягування до HIGH для уникнення неправильних сигналів */
  pinMode(switchPin6, INPUT); /* Встановлення піна для шостої кнопки як вхід */
  digitalWrite(switchPin6, HIGH); /* Підтягування до HIGH для уникнення неправильних сигналів */
  pinMode(soundPin, OUTPUT);  /* Встановлення піна для звукового сигналу як вихідний */
}

void loop()
{
  float h = dht.readHumidity();  /* Зчитування вологості з датчика DHT */
  float t = dht.readTemperature();  /* Зчитування температури з датчика DHT */

  //кнопка 1: секундомір
  if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
  {
    tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
    test=1; /* Вказуємо змінній test значення 1. Це робиться для запуску циклу, який буде відповідати за хід шкали таймера в реальному часі. */
    startTime = millis(); /* Ініціалізація таймера */

    while(test==1) /* Запускаємо цикл */
    {
      lcd.clear(); /* Чистемо інформацію на екрані дисплею */
      lcd.setCursor(0, 0); /* Вказуємо кординату дисплею для виведення інформації */

      unsigned long elapsedTime = millis() - startTime; /* Визначаємо час, який пройшов від початку таймера */
      int minutes = (elapsedTime % 3600000) / 60000; /* Обчислення хвилин */
      int seconds = (elapsedTime % 60000) / 1000; /* Обчислення секунд */
      int milliseconds = elapsedTime % 1000; /* Обчислення мілісекунд */

      /* Форматуємо час для виведення на екран */
      String secondsString = String(seconds < 10 ? "0" + String(seconds) : String(seconds)); /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String minutesString = String(minutes < 10 ? "0" + String(minutes) : String(minutes));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String millisecondsString = String(milliseconds < 10 ? "0" + String(milliseconds) : String(milliseconds));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */

      lcd.print("Elapsed Time:"); /* виводимо даний текст на верхню частину дисплея. Координати були вказані раніше. */
      lcd.setCursor(0, 1);  /* Вказуємо кординату дисплею для виведення інформації */
      lcd.print(String(minutesString) + "." + String(secondsString) + "." + String(millisecondsString)); /* виводимо пройдений час у іншій половині дисплею. */
      delay(400); /* Оновлюємо інформацію кожні 400 мілісекунд. */
```

```
      if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
        {
          tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
          test=0; /* Присвоюємо інше значення. По цій причині секундомір зупиняється (відбувається
вихід з циклу). */
          delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд. */
        }
      }
    }

    if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      selectedHour = (selectedHour + 1) % 24; /* Збільшення години на 1, обмеження до 23 (0-23 години).
Значення збільшується при кожному натисканні. */
      lcd.clear(); /* Очищення дисплею */
      String HourString = String(selectedHour < 10 ? "0" + String(selectedHour) :
String(selectedHour));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей
це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      lcd.setCursor(0, 0); /* Вказуємо координати */
      lcd.print("Hour: "); /* Виводимо текст */
      lcd.print(HourString); /* Виводимо значення вказаної раніше годинни */
      delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
    }

    if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      selectedMinute = (selectedMinute + 1) % 60; /* Збільшення хвилини на 1, обмеження до 59 (0-59
хвилин) */
      lcd.clear(); /* Очищення дисплею */
      String MinuteString = String(selectedMinute < 10 ? "0" + String(selectedMinute) :
String(selectedMinute));  /* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на
дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      lcd.setCursor(0, 0);  /* Вказуємо координати */
      lcd.print("Minute: ");  /* Виводимо текст */
      lcd.print(MinuteString); /* Виводимо значення вказаної раніше хвилини */
      delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
    }

    allMili = (selectedHour * 3600) + (selectedMinute * 60); /* Обчислення часу в секундах */
    lt = 1; /* Позначення, що таймер встановлено */

    if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четвертої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      while(lt == 1) /* Запускаємо цикл */
```

```
    {
      lcd.clear(); /* Очищення дисплею */
      int elapsedTime1 = allMili--; /* Зменшення часу */
      int hours = (elapsedTime1 % 86400) / 3600; /* Обчислення годин */
      int minutes = (elapsedTime1 % 3600) / 60; /* Обчислення хвилин */
      int seconds = (elapsedTime1 % 60); /* Обчислення секунд */

      String hoursString1 = String(hours < 10 ? "0" + String(hours) : String(hours));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String secondsString1 = String(seconds < 10 ? "0" + String(seconds) : String(seconds));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */
      String minutesString1 = String(minutes < 10 ? "0" + String(minutes) : String(minutes));/* Перевірка на число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку просто виводиться це значення. */

      lcd.setCursor(0, 0);  /* Вказуємо координати */
      lcd.print("Timer Time:"); /* Виводимо текст */
      lcd.setCursor(0, 1);  /* Вказуємо координати */
      lcd.print(String(hoursString1) + "." + String(minutesString1) + "." + String(secondsString1)); /* виводимо пройдений час таймеру у іншій половині дисплею. */
      delay(1000);  /* Оновлюємо інформацію кожну секунду */
      if (elapsedTime1 == 0) /* Перевірка на закінчення часу */
      {
        /* Генерація сигналу про закінчення таймера */
        tone(soundPin, 2000, 500); /* Відтворення звукового сигналу */
        delay(1000);  /* пауза на секунду */
        tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */
        delay(1000);  /* пауза на секунду */
        tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */
        delay(1000);  /* пауза на секунду */
        tone(soundPin, 2000, 500);  /* Відтворення звукового сигналу */

        lt = 0; /* Зупинка таймера */
        /* Обнулення всіх раніше введених значень. */
        elapsedTime1 = 0;
        selectedHour = 0;
        selectedMinute = 0;
        hours = 0;
        minutes = 0;
        seconds = 0;
        delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
      }

      if (digitalRead(switchPin4) == LOW) /* Перевірка на повторне натискання четвертої кнопки */
      {
        tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
```

```
      lt = 0; /* Зупинка таймера */
      /* Обнулення всіх раніше введених значень. */
      elapsedTime1 = 0;
      selectedHour = 0;
      selectedMinute = 0;
      hours = 0;
      minutes = 0;
      seconds = 0;
      delay(500); /* Оновлюємо інформацію кожні 500 мілісекунд */
    }
  }
}
if (digitalRead(switchPin5) == LOW) /* Перевірка на натискання п'ятої кнопки */
{
  tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
  lcd.clear(); /* Очищення дисплею */
  ns = 1; /* Вказуємо змінній ns значення 1. Це робиться для запуску циклу. */
  while(ns == 1) /* Запускаємо цикл */
  {
    if (isnan(h) || isnan(t))  /* Перевірка на помилку */
    {
      lcd.print("err"); /* вивід тексту */
      delay (3000); /* оновлювати кожні 3 секунди */
      lcd.clear(); /* Очищення дисплею */
      return;
    }
    else
    {
      lcd.setCursor(0, 0); /* вказуємо координати */
      lcd.print("temp: "); /* вивід тексту */
      lcd.print(t); /* вивід температури */
      lcd.print(char(223)); /* вивід знака градусів */
      lcd.print("C"); /* вивід тексту */
      lcd.setCursor(0, 1); /* вказуємо координати */
      lcd.print("hum: "); /* вивід тексту */
      lcd.print(h); /* вивід вологості */
      lcd.print(char(37)); /* вивід знака відсотків */
      delay (2000); /* оновлювати давнні кожні 2 секунди */
      lcd.clear(); /* вичищення дисплею */
      if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
      {
        ns = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки */
      {
        ns = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки */
```

```
          {
            ns = 0; /* вихід з циклу */
          }
          if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четверту кнопки */
          {
            ns = 0; /* вихід з циклу */
          }
          if (digitalRead(switchPin6) == LOW) /* Перевірка на натискання шостої кнопки */
          {
            ns = 0; /* вихід з циклу */
          }
        }
      }
    }
    if (digitalRead(switchPin6) == LOW) /* Перевірка на натискання шостої кнопки */
    {
      tone(soundPin, 800, 250); /* Відтворення звукового сигналу */
      lcd.clear(); /* Очищення дисплею */
    var = 1; /* значення для старту циклу */
    while(var == 1) /* запуск циклу */
    {
      RtcDateTime now = Rtc.GetDateTime();  /* Отримання поточної дати та часу з модуля RTC */
      int hour = now.Hour(); /* введеня години */
      int minute = now.Minute(); /* введення хвилини */
      int second = now.Second(); /* введеня секунди */
      int year = now.Year(); /* введеня року */
      int month = now.Month(); /* введення місяця */
      int day = now.Day(); /* введення дня */

      String hourString = String(hour < 10 ? "0" + String(hour) : String(hour));/* Перевірка на число,
а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку
просто виводиться це значення. */
      String minuteString = String(minute < 10 ? "0" + String(minute) : String(minute));/* Перевірка на
число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому
випадку просто виводиться це значення. */
      String secondString = String(second < 10 ? "0" + String(second) : String(second));/* Перевірка на
число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому
випадку просто виводиться це значення. */
      String monthString = String(month < 10 ? "0" + String(month) : String(month));/* Перевірка на
число, а саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому
випадку просто виводиться це значення. */
      String dayString = String(day < 10 ? "0" + String(day) : String(day));/* Перевірка на число, а
саме чи воно менше 10. Якщо так, ми виводимо на дисплей це число з 0 поперед собою. В іншому випадку
просто виводиться це значення. */

      String dateString = String(dayString) + "/" + String(monthString) + "/" + String(year); /*
розтановка значень для ввиводу */
      String timeString = String(hourString) + ":" + String(minuteString) + ":" + String(secondString);
```

```
/* розтановка значень для ввиводу */
      lcd.setCursor(0, 0); /* вибід координати */
      lcd.print("Date: "); /* вивід тексту */
      lcd.print(dateString); /* вивід дати */
      lcd.setCursor(0, 1); /* вибір координати */
      lcd.print("Time: "); /* вивід тексту */
      lcd.print(timeString); /* вивід часу */
      delay(1000); /* оновлювати щосекундно */
      lcd.clear(); /* Очищення дисплею */
      if (digitalRead(switchPin1) == LOW) /* Перевірка на натискання першої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin2) == LOW) /* Перевірка на натискання другої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin3) == LOW) /* Перевірка на натискання третьої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin4) == LOW) /* Перевірка на натискання четвертої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
      if (digitalRead(switchPin5) == LOW) /* Перевірка на натискання п'ятої кнопки */
      {
        var = 0; /* вихід з циклу */
      }
    }
  }
}
```