Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра комп'ютерних систем та технологій

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Побудова апаратно-програмного пристрою для розумного будинку»

«Creation of a hardware and software device fot a smart house»

Виконав: здобувач денної форми навчання
спеціальності 123 Комп'ютерна інженерія
Освітня програма «Комп'ютерна інженерія»

Лебедєв Ігор Сергійович
Керівник _д.т.н., проф. Гунченко Ю.О._____
(наук. ступінь, вчене звання, прізвище та ініціали, підпис)
Рецензент _к.ф.-мат.н., доц. Шугайло Ю.Б._____
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ 12 від «09» 06 2023 р.
Завідувач кафедри

_____ _Юрій ГУНЧЕНКО
(підпис) (прізвище, ім'я)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)
Голова ЕК

_____ __Алла КОБОЗЄВА___
(підпис) (прізвище, ім'я)

Одеса – 2023

Форма № Н-9.01

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І. МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

Кафедра комп'ютерних систем та технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 123-Комп'ютерна інженерія

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

проф. Ю. О. Гунченк

«___» _____ 2023 року

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Лебедєву Ігорю Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «побудова апаратно-програмного пристрою для розумного будинку».
   Керівник проекту (роботи) Гунченко Ю.О., д.т.н., проф.,
   (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
2. Строк подання студентом проекту (роботи)
   _____
3. Вихідні дані до проекту (роботи) _____
   _____
   _____
   _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   _____
   _____
   _____
   _____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
   _____
   _____
   _____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного пректу (роботи)	Строк виконання етапів проекту (роботи)	Примітка

Студент_____ _____

(прізвище та ініціали)

Керівник проекту (роботи) _____ _____

(прізвище та ініціали)

АННОТАЦІЯ

У дипломній роботі розробляється тема "Побудова апаратно-програмного пристрою для розумного будинку".

Мета роботи – розробка та складання електронного пристрою, який називається «годинник-будильник» і може використовуватися двома способами: як годинник і як будильник, при цьому обидві ці завдання можуть виконуватися одночасно. Складання цього пристрою відбувається на основі мікроконтролера Arduino Uno який управляє пристроєм, також в ньому використовуються й інші електронні елементи такі як інфрачервоний пульт та інфрачервоний приймач дистанційного керування, активний зумер, чотирирозрядний 7-сегментний світлодіодний індикатор, три тактові кнопки реального часу DS1302. З усіх вищезгаданих частин пристрою потрібно зібрати один цілий пристрій годинника-будильника який повинен визначати та враховувати час, виводити на чотиризначний 7-сегментний світлодіодний дисплей або актуальний час, або час, який користувач налаштовує для дзвінка будильника, або час який користувач хоче встановити на годинник замість того часу, який вже стоїть на цьому годиннику. Також цей пристрій повинен дзвонити в заздалегідь вказаний користувачем час і мати два функціонали налаштування: функціонал переведення годинника на інший час і функціонал налаштування часу дзвінка будильника. При цьому повинна бути можливість будь-яким з цих функціоналів скористатися або за допомогою тактових кнопок, або використовуючи пульт дистанційного управління.

ANNOTATION

In the thesis, the topic "building a hardware and software device for a smart home" is being developed.

The purpose of the work is the development and assembly of an electronic device called "alarm clock" and can be used in two ways: as a clock and as an alarm clock, while both of these tasks can be performed simultaneously. The assembly of this device is based on the Arduino Uno microcontroller that controls the device, it also uses other electronic elements such as an infrared remote control and an infrared remote control receiver, an active buzzer, a four-digit 7-segment LED indicator, three tact buttons with four pins each and a clock real time DS1302. From all of the above parts of the device, you need to assemble one whole alarm clock device that should determine and take into account the time, display on a four-digit 7-segment LED display or the current time, or the time that the user sets either as the alarm time, or as the time which the user wants to set on the clock instead of the time that is already on this clock. Also, this device should ring at a time specified by the user and have two setting functions: the function of setting the clock to another time and the function of setting the alarm time, while any of these two functions should be able to be used either using the tact buttons or using the remote control management.

ЗМІСТ

ВСТУП

Існує велика кількість різних видів діяльності, які виконуються протягом деякого визначеного проміжку часу, за який вони повинні бути виконані. Наприклад, люди можуть домовлятися про час зустрічі в тому чи іншому заздалегідь зазначеному місці, можуть працювати протягом деякого визначеного проміжку часу з однієї години по іншу, можуть мати годину фіксованої обідньої перерви або відпочинку на обмежений час, також існує величезна кількість інших справ тимчасовий проміжок виконання якого заздалегідь заплановано.

Для того, щоб виконувати ці дії за певний час і визначати залишок часу до того моменту, коли справу необхідно завершити, потрібен інструмент, дозволяючий визначати і враховувати час і його проміжки, за які людина повинна завершити її виконання. За цієї мети і було винайдено пристрій, який називається годинником.

Годинник буває різних видів, типів, кольорів, форм, розмірів і принципів дії, при цьому їх різноманітність дуже велика. За принципом дії годинники поділяються на сонячні, водяні, свічкові, пісочні, кварцові, механічні, пружинні, маятникові, атомні та електронні. При цьому годинники також діляться і на настільні, настінні і наручні, вони відрізняються між собою тим як і де їх зберігають і використовують, настінні висять на стіні, настільні лежать на столі або поличці шафи, а наручні носять на ремінці який застебнутий на руці. Якщо годинник має функцію дзвінка в заздалегідь вказаний користувачем час то тоді такий годинник називається будильником і використовується найчастіше для того, щоб прокинутися в обраний

користувачем час, для того щоб можна було вказати час коли будильник повинен задзвеніти цей годинник-будильник повинен мати функціонал налаштування часу будильника.

У моїй дипломній роботі я зберу з різних деталей настільний електронний годинник-будильник який має подвійний функціонал: його можна використовувати і як звичайний годинник, і як будильник.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Постановка завдання

У цій дипломній роботі опрацьовується тема «побудова апаратно-програмного пристрою для розумного будинку».

Вибір часу, коли задзвонить будильник, повинен бути можливий двома способами: за допомогою інфрачервоного пульта дистанційного керування і за допомогою тактових кнопок припаяних до цього пристрою і безпосередньо з'єднаних з мікроконтролером Arduino за допомогою з'єднувальних проводів-перемичок. Тактових кнопок має бути три, вони повинні бути припаяні поряд одна з одною в одну лінію. Перша тактова кнопка повинна зменшувати час, коли задзвенить будильник на 1 хвилину, друга повинна при натисканні вмикати або вимикати меню налаштування будильника в залежності від того, в якому стані знаходиться будильник на момент натискання цієї кнопки, а третя - повинна збільшувати час, коли задзвенить будильник на хвилину. При цьому якщо час дзвінка будильника, що налаштовується, дорівнює 23:59, то тоді в разі натискання кнопки додавання часу настроюваний час дзвінка будильника зміниться на 00:00, якщо час дзвінка будильника, що налаштовується, дорівнює 00:00, то тоді в разі натискання кнопки зменшення часу настроюваний час дзвінка будильника зміниться на 23:59.

Якщо настає момент, коли повинен задзвеніти будильник, тоді встановлений у будильник активний зумер-динамік починає дзвініти, і

дзвінітиме доки користувач не вимкне будильник. Це можливо двома способами:

а) Використовуючи інфрачервоний пульт дистанційного керування.

б) Натиснувши розташовану між двома іншими тактовими кнопками кнопку, яка припаяна до будильника.

Актуальний час (або час, що настроюється, якщо пристрій знаходиться в настроювальному меню) повинен відображатися на чотиризначному семисегментному LED дисплеї у вигляді: «години.хвилини».

Крім цього повинна бути можливість переведення годинника на інший час у разі необхідності. Вона передбачена, тому що може бути необхідна у наступних випадках:

а) Коли користувач хоче перевести годинник із зимового часу на літнє.

б) Коли користувач хоче перевести годинник із літнього часу на зимове.

в) Коли батарея сіла та годинник деякий час поспіль був у неробочому стані, то користувач після зміни батареї на робочу повинен буде встановити на годинник правильний час.

г) У випадку міжнародних переміщень, якщо людина опиняється в іншому часовому поясі. Користувачеві доведеться перевести пристрій обліку часу на кількість годин, що дорівнює тимчасовій різниці цих двох часових поясів.

Переведення часу на годиннику має бути можливим двома способами: використовуючи пульт дистанційного керування та за допомогою трьох тактових кнопок, які використовуються і для налаштування часу, коли задзвонить будильник і для зміни часу, який стоїть на годиннику. Перший спосіб використання тактових кнопок був описаний вище, а другий спосіб наступний: перша тактова кнопка повинна при натисканні включати або вимикати меню налаштування годинникового механізму в залежності від поточного стану годинника на момент натискання цієї кнопки, друга повинна

зменшувати час, який ми встановлюємо на годиннику на 1 хвилину, а третя - повинна збільшувати час, який ми встановлюємо на годиннику на 1 хвилину. При цьому якщо час, який стоїть на годинниковому механізмі дорівнює 23:59, то тоді у разі натискання кнопки додавання часу час на годиннику зміниться на 00:00, якщо час, який стоїть на годинниковому механізмі дорівнює 00:00, то тоді у разі натискання кнопки зменшення часу час на годиннику зміниться на 23:59.

Управління цим пристроєм за допомогою інфрачервоного пульта здійснюється поетапним натисканням необхідної послідовності кнопок у наступному порядку:

а) Спочатку вибираємо необхідну дію (кнопки: «1» – налаштування часу, коли має задзвеніти будильник, «2» – зміна часу на годиннику (переведення годинника), «3» – вимикання дзвінка активного зумера, який починає дзвеніти у встановлений на будильнику користувачем час (після того, як вона натиснута будильник автоматично вимикається)).

б) Налаштування часу, коли задзвенить будильник (якщо до цього була натиснута кнопка «1») або часу, що стоїть на годиннику (якщо до цього була натиснута кнопка «2») здійснюється за допомогою кнопок «-» (якщо треба відняти хвилини, одне натискання цієї кнопки зменшує час, що настроюється користувачем на 1 хвилину) і «+» (якщо треба додати хвилини, одне натискання цієї кнопки збільшує час, що налаштовується користувачем, на 1 хвилину).

в) Вихід із меню налаштування часу дзвінка будильника відбувається тоді, коли користувач натискає кнопку «4», а вихід із меню налаштування часу, що стоїть на годиннику (тобто з меню переведення часу на годинниковому пристрої) здійснюється натисканням кнопки «5».

Пристрій має бути зібраний на спеціальній паяльній платі, яка повинна бути основою до якої припаюються інші елементи (плата Arduino Uno, приймач інфрачервоного сигналу, чотирирозрядний 7-сегментний індикатор, три тактові кнопки чотирипіни, годинник реального часу DS1302). З'єднання

різних компонентів пристрою з мікроконтролером має відбуватися за допомогою з'єднувальних дротів. Пізніше зібрана схема має бути вставлена в пластмасовий корпус для годинника і закріплена в ньому.

1.2 Короткий опис пристрою

1.2.1 Історія виникнення та розвитку годинників

Годинник – це спеціальний пристрій для відліку, обліку та визначення часу. Винайдені вони були в давнину, і історія їх розвитку від появи до наших днів налічує тисячоліття. Найдавніший годинник був сонячним і в залежності від положення сонця вдень по ньому визначали час. Спочатку в якості сонячного годинника використовували палицю яку встромляли в землю і залежно від положення тіні та її розміру отримували дані про актуальний час дня. Саме цей посох став першим і найпростішим елементом сонячного годинника, який називається гномоном, і саме на його основі надалі відбувався розвиток та еволюція сонячного годинника, який вказував актуальний час залежно від положення сонця на небосхилі. Пізніше було виділено коло навколо устромленого в землю палиці і ще пізніше його розділили на сегменти рівні між собою і дозволяють точніше визначати час. Вперше це коло сонячно-місячного годинника було поділено на 60 сегментів у давній державі Шумеро-Аккадського царства приблизно в 2000 році до нашої ери. Розподіл часового кола сонячно-місячного годинника на два рівні дванадцятисегментних проміжків часу, сегменти яких у свою чергу рівні між собою, вперше було придумано в стародавньому Єгипті, де визначення часу відбувалося за допомогою великих обелісків-гномонів, які використовувалися для визначення та обліку актуального часу доби. як раніше при цьому використовувалися палиці [1].

Протягом тисячоліть було створено велику кількість різних моделей сонячного годинника. Дуже довго він був основним способом визначення та обліку часу і приніс велику користь людству. Але сонячні годинники мали серйозний недолік: вони працювали тільки в тому випадку якщо яскраво світило сонце. Вони ставали марними, якщо не було сонця (наприклад вночі чи за поганої погоді) їх використання, а також визначення і облік часу ставало неможливим. Також їх було незручно використовувати ще й тому, що вони вимагали постійного налаштування свого положення при кожній зміні пори року, причиною чого була нерівномірність руху сонця небесним склепінням [1].

Майже що одночасно з сонячним годинником у стародавньому Єгипті більш ніж за 2000 років до нашої ери був придуманий годинник іншого виду-водний. Незабаром після того, як вони були винайдені, вони набули широкого поширення, спочатку їх запозичили народи Месопотамії, а з Месопотамії їх запозичили китайці, що жили в древньому китайському царстві Шан і використовували їх у той же час, як і єгиптяни [1].

У стародавньому Перському царстві починаючи з 2500 рр. до н.е. для визначення актуального часу доби повсюдно використовувався водяний годинник. Крім того, для вимірювання часу використову, валися ще й свічкові годинники за допомогою яких час визначався по тому яка частина свічки встигла згоріти за проміжок часу від того моменту, коли по цьому годиннику було розпочато відлік часу. Їх використовували народи Китаю, Японії, Англії та Ірану. Сонячний же годинник Індії та Тибету мав дуже незвичайну форму: він складався просто зі спеціально підібраної для цих цілей палиці [1].

Починаючи з 3 століття до н.е. в грецьких літературних джерелах з'являються перші згадки про винайдений у Стародавній Греції годинник з водяним спусковим механізмом, який перетворював обертальну енергію на переривчастий рух. Ще пізніше, в 10 столітті, на території стародавнього Китаю був винайдений годинник, де використовувався вже не водяний

спусковий механізм, а ртутний. Ще пізніше арабські майстри вдосконалили водяний годинник додавши механічні передачі необхідні для перетворення крутних моментів [1].

У 14 столітті були винайдені механічні годинники в яких використовується штирьовий спусковий механізм і довгий час вони були основним видом пристрою для визначення та обліку часу. Трохи пізніше ніж механічний годинник з'явився пружинний годинниковий механізм і кишеньковий годинник який був винайдений у 16 столітті. Незабаром після появи пружинного і кишенькового годинника було винайдено маятниковий годинник, який більш ніж 300 років поспіль був найточнішим годинником з усіх існуючих на той час. У минулому столітті був винайдений кварцовий і трохи пізніше атомний годинник, який на даний момент є найточнішою моделлю годинника з усіх, які зараз існують і тому саме з ними звіряються всі інші види годинників і лежать в основі всесвітнього координованого часу [1].

1.2.2 Опис пристрою

Пристрій годинник-будильник складається з наступних частин:

а) Плати Arduino Uno R3 CH340, яка забезпечує керування пристроєм і керує роботою інших частин, з яких складаються годинник-будильник і які виконують свої особливі функції при роботі цього пристрою.

б) Інфрачервоних приймача та пульта дистанційного керування, які надають можливість дистанційного керування годинником реального часу DS1302 або при переведенні часу, встановленого на годиннику, або при налаштуванні часу дзвінка будильника.

в) Активного зумера, який використовується для виробництва звукових сигналів, які має видавати цей пристрій, коли настає час, вказаний користувачем як час, коли будильник повинен почати дзвеніти.

г) Чотирирозрядний 7-сегментний індикатор який використовується для відображення або настроюваного, або актуального часу.

д) Три тактові кнопки з чотирма пінами, які використовуються при ручному керуванні годинником реального часу DS1302 або при переведенні часу, встановленого на годиннику, або при налаштуванні часу дзвінка будильника.

е) Годинник реального часу DS1302 RTC, який використовується для визначення та обліку актуального часу, а також надає дані, що дозволяють визначити момент настання часу, який користувач вибрав як час, коли будильник повинен почати дзвонити.

Ці 6 частин пристрою складають один цілий пристрій який називається «годинник-будильник» і використовується двома способами:

а) Як годинник реального часу.

б) Як будильник, який повинен дзвеніти в заздалегідь вказаний користувачем час.

Цей годинник є електронним і призначений для домашнього використання, найкращим місцем їх встановлення є велика рівна площина, якою може бути або полиця шафи, або стіл.

1.2.3 Основні особливості електронного годинника

Електронний годинник є найбільш поширеним у наш час видом годинника, його використовують і як окремий пристрій (у наручному та настінному годиннику) і як частину інших пристроїв (їх вставляють у меблі, побутові прилади та транспортні засоби), основними причинами їх високої популярності є менша вартість і більша точність електронного годинника в порівнянні з механічним годинником.

1.2.4 Застосування електронного годинника

З тих пір як були винайдені перші електронні годинники, компанії, які займаються їх виробництвом, в результаті конкуренції між собою встигли сильно вдосконалити їх архітектуру і функціонал. На даний момент кількість їх моделей і видів величезна, вони використовуються скрізь і всюди і на

даний момент більшість годинників, що використовуються людьми. - Електронні. Електронні годинники за типом застосування діляться на вбудовані, стаціонарні та наручні.

Вбудований годинник - це годинник, який вставляють в різноманітну побутову техніку, меблі або в автомобілях поряд з кермом. Прикладом вбудованого годинника можуть бути сучасна духова шафа, пральні та посудомийні машини і мікрохвильові печі. Їх транспортування з одного місця в інше здійснюється як правило разом із тією побутовою технікою, у яку вони вбудовані.і

Стаціонарний годинник – це такий годинник, який є або настінним або годинником-будильником, він не вбудований в будь-яке обладнання і може без особливих проблем переноситися з одного місця в інше, але через свої розміри незручний для постійного перенесення, тому його не носять із собою. постійно але тільки у виняткових випадках, наприклад, при переїзді в інше місце проживання, при їх купівлі або продажу, а також при необхідності користувач може віднести їх у ремонт.

Наручний годинник - це годинник, який користувач постійно носить із собою, він зручний для носіння на руці і, як правило, до них приробленій спеціальний ремінець, який фіксує їх положення на руці користувача. Цей вид годинника найрізноманітніший з усіх, у ньому часто зустрічаються різні додаткові функції, наприклад вони можуть бути не тільки годинниками-будильниками, але в них можуть бути додані ще й компас, ліхтарик або ще якісь додаткові функції. До годинника цього виду пред'являються підвищені вимоги щодо міцності, водонепроникності та водостійкості, тому що саме їх беруть із собою в туристичні поїздки та походи, на спортивні заходи, на рибалку, на полювання та на екскурсії, тобто в місця, де користувач виявляє найбільшу фізичну активність і де вони можуть ударятися, падати , дряпатися, заливатись водою і піддаватися іншим подібним випробуванням на міцність.

Годинник-будильник, який я сконструював під час виконання дипломної роботи, є стаціонарними і можуть використовуватися користувачем двома способами: по перше, як годинник реального часу в завдання якого входить визначення та облік реального часу; а по друге, як будильник телефонуючого в заздалегідь налаштований користувачем час. Користувач може переводити цей годинник на інший час або встановлювати будильник на потрібний користувачеві час, коли він повинен задзвеніти.

1.3 Середа розробки Arduino IDE

Управління електронним пристроєм проводиться спеціально написаною програмою-драйвером, яка містить опис поведінки пристрою за різних умов його функціонування.

Пишуться вони у спеціальних інтегрованих середовищах розробки, вибір однієї з яких для використання залежить від моделі використовуваного мікроконтролера. Наприклад, я використовую мікроконтролер Arduino Uno R3 CH340 тому драйвер до нього буде написано в Arduino IDE, яка є спеціальним інтегрованим середовищем, створеним спеціально для мікроконтролерів, що випускаються компанією Arduino.

Інтегроване середовище розробки забезпечує всі необхідні функції та механізми для написання драйверів для електронних пристроїв і використовується для написання програм управління різними електронними пристроями, що створюються користувачем.

Інтегроване середовище розробки Arduino IDE підтримує лише мови Сі та C++ та має ряд своїх особливостей у тому числі:

а) Особливу структуру програми драйвера, що містить у собі два обов'язкові блоки методів setup і loop які використовуються наступним чином: метод setup містить у собі визначення режимів роботи висновків,

ініціалізацію змінних, запуск бібліотек, що використовуються, і т.д. а метод loop містить у собі опис поведінки електронного пристрою за різних умов.

б) Особливу послідовність виконання програми, в якій спочатку виконується один раз метод setup, а потім починається нескінченне циклічне виконання методу loop, при цьому можливе визначення та використання інших методів, посилання на які можуть бути вставлені в код, що знаходиться всередині методу loop, при цьому якщо посилання на інші методи зустрічається всередині цього методу відбувається перехід в блок цього методу і починається виконання що знаходиться в ньому коду після виконання якого відновлюється виконання методу loop починаючи з тієї команди на якій його виконання було перервано для виконання методу посилання на який була зустрінута в цьому місці коду у методі «loop».

У ній можна також використовувати раніше певні методи та константи, які об'єднані в бібліотеки, які можна підключити за допомогою «using».

Приклад головного меню програми Arduino IDE наводжу на рис. 1.1.

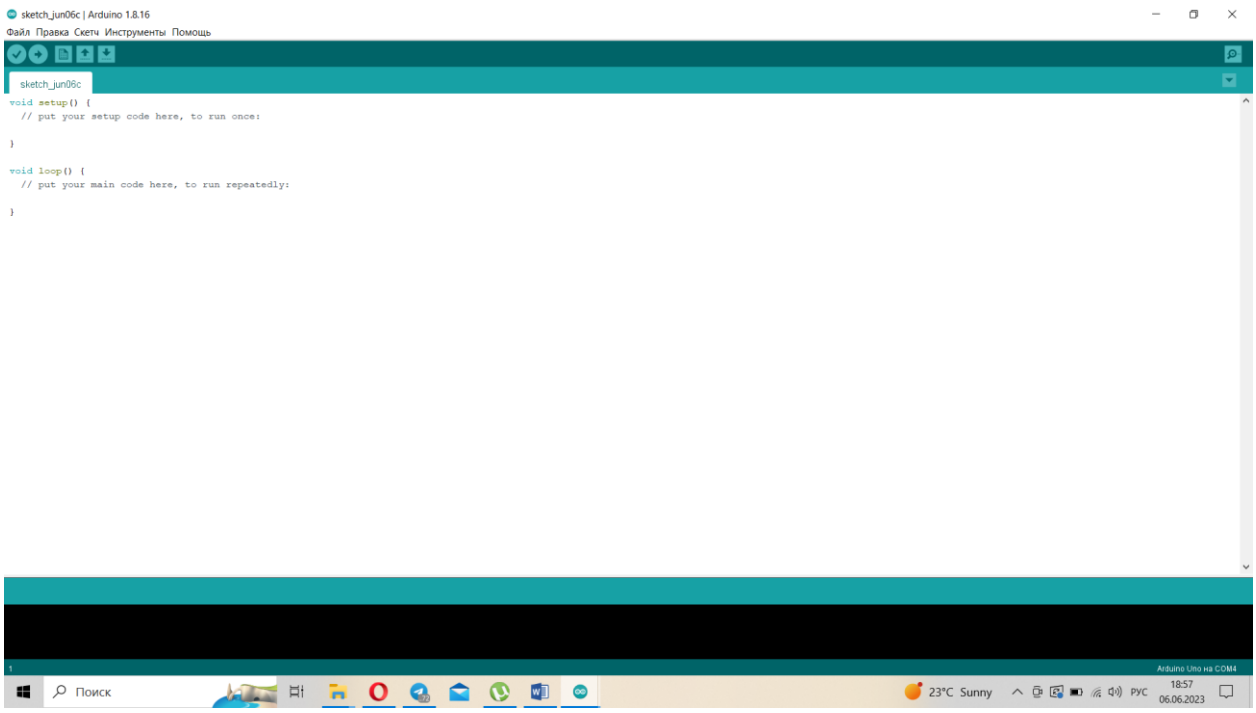

Рисунок 1.1 – Програма Arduino IDE

2 ОПИС КОМПОНЕНТІВ ЕЛЕКТРОНОГО ПРИСТРОЮ «ГОДИННИК-БУДИЛЬНИК»

2.1 Плата Arduino Uno R3 CH340 опис та характеристики

2.1.1 Опис плат Arduino Uno

Плата Arduino Uno R3 CH340 – це спеціальний мікроконтролер призначений для керування різними електронними пристроями, які на його основі можуть бути зібрані користувачем бажаючим зібрати будь-який корисний пристрій, причому кількість варіантів пристрою, який може зібрати користувач величезно, а можливість складання пристрою має тільки одне обмеження – фантазію користувача розробляючого пристрій, використовуючи цю плату можна зібрати будь-який пристрій, який захочеш зібрати.

У цьому проекті я зібрав електронний пристрій, який називається «годинник-будильник».

Для того щоб пристрій працював належним чином повинні бути дотримані дві умови:

а) Усі складники пристрою повинні бути правильно зібрані і з'єднані з мікроконтролером або між собою.

б) У мікроконтролер Arduino Uno R3 CH340 має бути завантажена правильно написана програма-драйвер пристрою, яка визначає поведінку зібраного пристрою за різних умов функціонування цього пристрою.

У разі виконання обох умов користувач розробник отримає правильно працюючий електронний пристрій власного складання, який він зможе використовувати для тих цілей, для яких він його розробляв і збирав. Наприклад, у моєму випадку годинник-будильник повинен використовуватися для обліку, відліку та визначення часу, а також для того, щоб у заздалегідь вказаний користувачем час робити звукові шуми з метою оповіщення, що цей час настав.

Нижче наводжу опис особливостей плати Arduino Uno R3 CH340:

а) Ця модель мікроконтролера має 14 цифрових та 6 аналогових виведень, які можна використовувати для керування різними елементами мікросхеми.

б) Цим мікроконтролером управляє процесор ATmega328P, що є частиною плати Arduino і припаяний до неї, крім того, цей мікроконтролер має 32 кілобайти флеш пам'яті.

в) Ця модель мікроконтролера вимагає від 5 до 12 вольт вхідного живлення.

г) Розмір цієї плати: 68*53*15 мм.

д) Цей мікроконтролер підтримує такі інтерфейси: I2C/TWI, SPI, PWM.

Для написання драйверів для пристроїв, що збираються користувачем використовується спеціальне інтегроване середовище розробки - Arduino IDE яка забезпечує всі необхідні інструменти для написання драйвера: вона дозволяє написати в ній код, вона його відкомпілює, завантажить в процесор мікроконтролера ATmega328P і видасть повідомлення про те, що драйвер завантажений в мікроконтролер та пристрій готовий до використання.

Зовнішній вигляд плати Arduino Uno R3 CH340 наводжу на рис. 2.1.

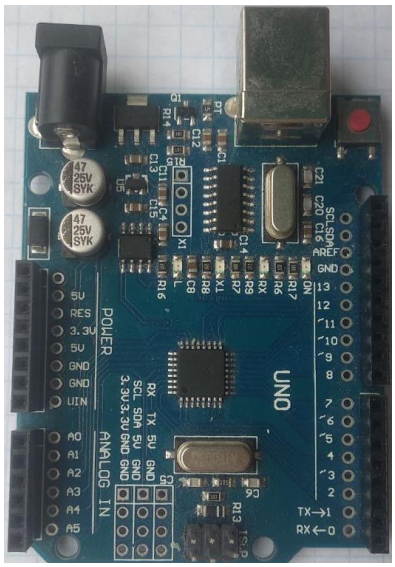

Рисунок 2.1 – Плата Arduino Uno R3 CH340

2.1.2 Використання плат Arduino Uno

Мікроконтролери використовуються у всіх електронних пристроях: від найпростіших до найскладніших, вони є обов'язковим компонентом будь-якої робочої схеми пристрою, тому що їх завдання – керування роботою електронного пристрою і це завдання здатні виконувати тільки мікроконтролери і ніякі інші пристрої виконувати не здатні. По суті мікроконтролери можна порівняти з мозком людини, саме вони відповідають за виконання програми драйвера яка визначає функціонування електронного пристрою за різних умов, саме вони визначають чи дотримуються ті чи інші умови, прописані в драйвері для виконання тієї чи іншої дії, саме вони забезпечують виконання циклів закладених Для виконання в програмі пристрою, саме вони виконують будь-які інші дії, закладені в програму пристрою.

Для того щоб електронний пристрій правильно працював необхідно правильно з'єднати всі його елементи так як якщо хоча б один проводок буде неправильно підключений або взагалі не буде підключений то це призведе до того, що весь пристрій буде працювати не так як хотів би користувач, тобто можливо 2 випадки:

а) Пристрій може взагалі не працювати.

б) Пристрій може працювати неправильно, у цьому випадку можливо два варіанти:

1) Можуть працювати всі інші частини пристрою крім тієї, де неправильно було здійснено підключення проводів (це якщо немає інших частин пристрою які залежать від працездатності цієї його частини).

2) Може не працювати неправильно підключена деталь пристрою і всі інші частини цього пристрою, що залежать від неї (це якщо є інші частини які залежать від працездатності цієї його частини).

Саме тому дуже важливо підключати інші елементи пристрою до мікроконтролера правильно, щоб підключення було правильним необхідно дотримуватися таких умов:

а) GND висновки інших елементів схеми необхідно підключати до одного із трьох GND висновків цього мікроконтролера або безпосередньо або через макетну плату;

б) VCC висновок (або висновок 5V) - це виведення живлення елементів схеми, через нього отримують живлення інші частини пристрою, його необхідно підключати до висновків інших елементів пристрою через їх 5V висновки (або висновки VCC) або безпосередньо або через макетну плату. Усього в платі Arduino Uno є 2 штуки VCC висновків номіналом 5V призначених для електроживлення інших елементів схеми.

в) Висновки різних елементів пристрою, призначені для управління ними, треба підключати або до цифрових висновків 0-13 або до аналогових висновків A0-A5 залежно від того, який елемент пристрою підключається, оскільки різні елементи підключаються до різних типів висновків, наприклад, потенціометр або датчики треба підключати до аналогового виводу, а світлодіоди або динаміки можна підключити до цифрових.

При дотриманні цих умов пристрій можна вважати зібраним правильно і якщо драйвер завантажений в мікроконтролер, написаний правильно, то і пристрій працюватиме правильно.

2.1.3 Висновок щодо плат Arduino Uno

Мікроконтролер Arduino Uno R3 CH340 є простим і зручним варіантом мікроконтролера, що дозволяє навіть людині без глибоких знань у галузі комп'ютерної схемотехніки проектувати та збирати з різних елементів різні корисні електронні пристрої. Ця модель мікроконтролера володіє хорошими технічними якостями, при цьому є однією з кращих плат, призначених для вивчення схемотехніки і основ робототехніки розробниками електронних пристроїв. Таким чином, вона є одним з кращих варіантів вибору для недосвідчених користувачів, бажаючих почати вивчення комп'ютерної схемотехніки і найпростішої робототехніки.

2.2 Принцип роботи IR пульта та IR приймача

У багатьох сучасних пристроях є необхідність у можливості дистанційного керування різними електронними пристроями, оскільки це набагато зручніше, ніж ручне керування за допомогою тактових кнопок. Для забезпечення можливості дистанційного управління використовуються інфрачервоні пульт та приймач дистанційного управління, принципи їх роботи будуть розглянуті в цьому розділі.

2.2.1 Передавач IR сигналу

Передавач IR сигналу (також називається ще й IR пультом) зазвичай отримує живлення від батарейки або акумулятора. Саме тому кількість споживаної ним електрики має бути якнайменшою. При цьому сигнал, який він випромінює, повинен бути досить потужним для забезпечення необхідної відстані передачі сигналу. Настільки різні за енергетичними витратами завдання успішно вирішуються способом передачі коротких кодованих імпульсних пакетів. У проміжках між передачами IR сигналу пульт дистанційного керування майже споживає електроенергію. Завдання мікроконтролера пульта – це зчитування сигналів з кнопок, вставлених у цей пульт, а також кодування інформації, модулювання опорної частоти та

видача сигналу на IR випромінювач. Для виготовлення пультів дистанційного керування випускаються не тільки різні спеціалізовані мікросхеми, але для цього можуть бути використані і сучасні мікроконтролери загального застосування типу AVR або PIC. Основна вимога до таких мікроконтролерів – це наявність режиму сну з мінімальним споживанням електроенергії та здатністю відчувати натискання кнопок у цьому стані. Випромінювач IR сигналу може випромінювати інфрачервоні промені в момент дії струму збудження. Струм на випромінювачі зазвичай перевищує можливості мікроконтролера, тому для формування необхідного струму встановлюється найпростіший світлодіодний драйвер на одному транзисторі. З метою зниження втрат потрібно при виборі транзистора звертати увагу на коефіцієнт посилення струму - β або h21. Чим вищий цей коефіцієнт, тим вища ефективність пристрою. Сучасні передавачі використовують польові або CMOS транзистори, ефективність яких на частотах є максимальною [2].

Описана вище схема має ряд недоліків, у тому числі і той недолік, що в міру зниження рівня заряду батареї потужність випромінювання теж падатиме, що врешті-решт призведе до зниження дальності передачі сигналів. Для зниження залежності від напруги живлення можна використовувати найпростіший стабілізатор струму [2].

Більшість пультів дистанційного керування працює на частоті 30 – 50 кГц. Цей діапазон частот було вибрано історично ще в ті часи, коли з'явилися перші подібні пристрої. Ця область була обрана тому що рівень перешкод у ній мінімальний. Також були прийняті в розрахунок та обмеження на елементну базу. Пізніше в міру стандартизації та розповсюдження апаратури з цим способом управління здійснення переходу на інші частоти стало недоцільним [2].

З метою збільшення імпульсної потужності передавача, а, отже, і дальності передачі інфрачервоного сигналу, сигнал основної частоти відрізняється від меандру і має шпаруватість 3 - 6. Таким чином

підвищується імпульсна потужність зі збереженням або навіть зменшенням середньої потужності. Імпульсний струм світлодіода вибирається, виходячи з його паспортних значень і може досягати одного і більше Ампер. Імпульсний струм у більшості IR пультів не перевищує 100 мА. Так як опорна частота має низький коефіцієнт заповнення і тривалість кодованої посилки не перевищує 20-30 мс, то середній струм при натиснутій кнопці не повинен перевищувати одного міліампера. Підвищення імпульсного струму світлодіода пов'язане зі зниженням ефективності та зменшенням терміну служби. Сучасні інфрачервоні світлодіоди мають ефективність 100-200 мВт випромінюваної енергії за струму 50 мА. Допустимий середній струм не повинен перевищувати 10-20 мА. Живлення світлодіода повинне мати RC фільтр, який знижує вплив імпульсної перешкоди на живлення мікроконтролера. Спектр світлодіодів для IR пультів більшості моделей побутової техніки має максимум в області 940 нм [2].

Тривалість одиничного пакета опорної частоти для впевненого прийому становить щонайменше 12-15 і трохи більше 200 періодів. При передачі кодованої посилки передавач формує спочатку преамбулу яка є один або кілька пакетів опорної частоти і дозволяє приймачеві встановити необхідний рівень посилення та фону. Дані кодованої посилці передаються у вигляді нулів і одиниць, які визначаються тривалістю або фазою (відстанню між сусідніми пакетами). Загальна тривалість кодованої посилки найчастіше становить від кількох біт до кількох десятків байт. Порядок прямування, ознака початку та кількість даних визначається форматом посилки [2].

Зовнійшній вигляд інфрачервоного пульту дистанційного керування Arduino наводжу на рис. 2.2.

Рисунок 2.2 – Інфрачервоний пульт дистанційного керування Arduino

2.2.2 Приймач IR сигналу

Приймач IR сигналу зазвичай містить у собі, крім самого приймача інфрачервоного випромінювання, ще й мікроконтролер. Мікроконтролер дешифрує отримуваний сигнал та виконує інші необхідні дії. Так як приймач найчастіше використовується в пристроях, що отримують живлення від мережі, тому його використання не дуже популярне. Крім того, мікроконтролер досить часто виконує інші особливі функції в пристрої і є його центральним логічним елементом [2].

Приймач IR випромінювання зазвичай виготовляється у вигляді окремого інтегрального модуля, який встановлюється за передньою панеллю керованого пристрою. У передній панелі є прозоре для інфрачервоних променів віконце. Найчастіше така мікросхема має три висновки: виведення живлення, загальний висновок та виведення виходу сигналу. Компанії-виробники електронних пристроїв вставляють у свої вироби різні види приймачів IR сигналів, які можуть відрізнятися типом і виконанням. Проте принцип їхньої роботи схожий. Усередині такої мікросхеми знаходяться:

а) Підсилювач, що інтегрує та виділяє корисний сигнал на рівні фону.

б) Обмежувач, який наводить сигнал до логічного рівня.

в) Фотоприймач-фотодіод.

г) Смужний фільтр, налаштований на частоту передавача.

д) Демодулятор-детектор, який виділяє огинаючу корисного сигналу [2].

Корпус такого приймача виготовлений з матеріалу, який виконує роль додаткового фільтра, що пропускає інфрачервоні промені певної довжини хвилі. Сучасні інтегральні приймачі дають можливість приймати корисний сигнал лише на рівні фону, що перевищує їх у кілька десятків разів і навіть відчувати посилки частоти, мають всього 4-5 періодів [2].

Живлення приймача випромінювання має відбуватися за допомогою RC фільтра, тому що так можна збільшити чутливість. Мікроконтролер справляє перешкоду широкого спектру на лініях живлення, що може вплинути на роботу приймача [2].

Зовнішній вигляд інфрачервоного приймача VS1838B наведено на рис.2.3.

Рисунок 2.3 – Інфрачервоний приймач VS1838B

2.3 Активний зумер

Активний зумер – це спеціальний пристрій, що використовується для виробництва звуків, від пасивного зумеру він відрізняється тим, що його не треба додатково налаштовувати, оскільки він уже містить в собі власний генератор звуку, який завжди генерує одну і ту ж звукову мелодію, яку змінити не можна, як у пасивному. Зумер її змінювати можна, але в той же

час цей вид зумер має і свій власний недолік - він вимагає обов'язкового налаштування видаваємого ним звуку перед використанням.

У проекті «годинник-будильник» я використовуватиму активний зумер для того, щоб у вказаний користувачем час будильник міг задзвеніти. Наразі дзвеніти він буде доти, доки користувач не вимкне його, натиснувши одну з двох кнопок: або натиснувши кнопку яка припаяна до пристрою між двома іншими тактовими кнопками чи натиснувши кнопку «3» на інфрачервоному пульті дистанційного керування.

Зовнішній вид зверху активного зумера наведено на рис. 2.4, вид знизу на рис. 2.5.

Рисунок 2.4 – Активний зумер вид зверху

Рисунок 2.5 – Активний зумер вид знизу

2.4 Чотирирозрядний 7-сегментний індикатор

2.4.1 Використування чотирирозрядного 7-сигментного індикатору

Семисегментний чотирирозрядний індикатор використовується коли треба вивести на дисплей інформацію цифрового формату і є одним із способів візуального відображення інформації з плати Arduino Uno у доступному для користувача вигляді, найчастіше його використовують в електронному годиннику та будильниках різних видів для відображення часу, в холодильниках, термометрах та градусниках для відображення

температури, барометрах для відображення атмосферного тиску, в далекомірах і маршрутизаторах для відображення відстані а також у багатьох інших подібних пристроях де є необхідність відображення числових значень на зовнішніх дисплеях.

Сам по собі цей дисплей складається з набору світлодіодів, які вставлені всередину пластмасового корпусу і об'єднані в чотири групи в кожну з яких входить цифра і точка, що знаходиться поруч з нею, знизу цього дисплея знаходиться 12 цифрових виведень, 7 з яких керують певними окремими світлодіодами обраної цифри, 1 виведення керує точкою поруч із цією цифрою, а 4 інші виводи використовуються для вибору якоїсь із чотирьох цифр, які входять до складу цього дисплея. Більш детально тему розпинування чотиризначних 7-сегментних індикаторів буде викладено та роз'яснено в наступному розділі.

Вид зверху чотиризначного 7-сегментного індикатору наведено на рис. 2.6.

Рисунок 2.6 – Чотиризначний 7-сегментний індикатор зверху

Вид зверху чотиризначного 7-сегметного індикатору наведено на рис. 2.7.

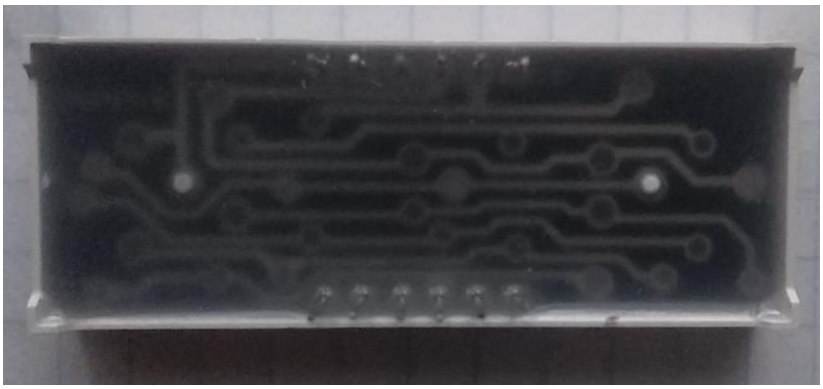

Рисунок 2.7 – Чотиризначний 7-сегментний індикатор знизу

2.4.2 Розпинування чотирирозрядного 7-сегментного індикатора

Тут D1 – вивід, що забезпечує вибір першої цифри, D2 – вивід, що забезпечує вибір другої цифри, D3 – вивід, що забезпечує вибір третьої цифри, D4 – вивід, що забезпечує вибір четвертої цифри, a a – вивід, що забезпечує вибір верхнього світлодіода, f – вивід, що забезпечує вибір верхнього лівого світлодіода. , b - вивід забезпечує вибір верхнього правого світлодіода, e - вивід забезпечує вибір нижнього лівого світлодіода, d - вивід забезпечує вибір нижнього світлодіода, dp - вивід забезпечує вибір світлодіода точки, c - вивід забезпечує вибір нижнього правого світлодіода, g - вивід забезпечує вибір центрального світлодіоду.

Для того щоб встановити на дисплей числове значення, треба виконувати необхідні дії в наступному порядку:

а) Вибрати цифру певного розряду числового індикатора, на якій встановлюватиметься значення.

б) Встановити необхідне значення на вибрану цифру потрібного розряду.

Для того, щоб встановити числові значення на всі чотири розряди цього чотиризначного семисегментного світлодіодного дисплея, треба повторити описаний вище алгоритм дій 4 рази.

Можливо 4 варіанти вибору числового розряду на 4-значному 7-сегментному індикаторі і проводиться вибір одного з цих варіантів наступним чином:

а) Щоб вибрати першу цифру треба подати на висновок D1 значення LOW, але в висновки D2, D3, D4 значення HIGH.

б) Щоб вибрати другу цифру треба подати на висновок D2 значення LOW, але в висновки D1, D3, D4 значення HIGH.

в) Щоб вибрати третю цифру треба подати на висновок D3 значення LOW, але в висновки D1, D2, D4 значення HIGH.

г) Щоб вибрати четверту цифру треба подати на висновок D4 значення LOW, але в висновки D1, D2, D3 значення HIGH.

Встановлення числового значення на цифру обраного розряду здійснюється за допомогою включення певної кількості необхідних для цієї мети світлодіодів, об'єднаних на чотиризначному семисегментному світлодіодному індикаторі в цифру потрібного нам розряду. Для увімкнення світлодіода цієї цифри треба подати на нього значення HIGH.

Зовнішній вигляд розпинки чотиризначного 7-сегментного світлодіодного індикатору наводжу на рис. 2.8

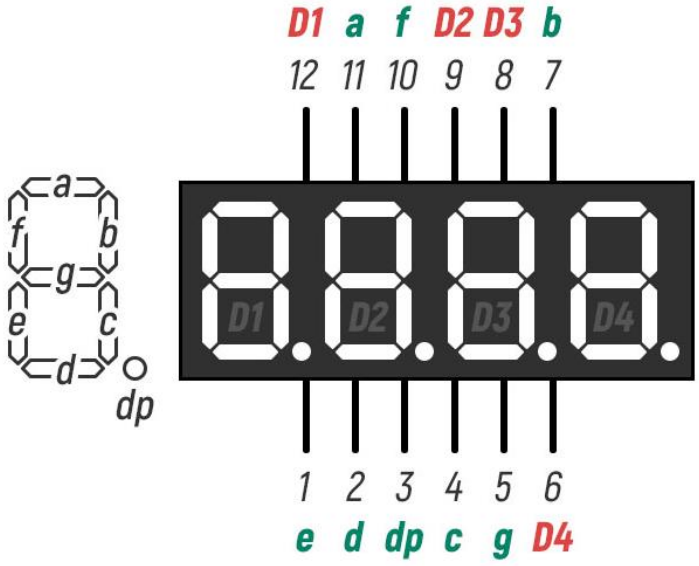

Рисунок 2.8 – Розпинка чотиризначного 7-сегментного світлодіодного індикатора

2.5 Тактова кнопка з чотирма пінами

Тактова кнопка – це спеціальний механізм, призначений для взаємодії між людиною, що її натискає, і яким-небудь електронним пристроєм з платою Arduino якого з'єднана ця тактова кнопка. Взаємодія між користувачем та мікроконтролером, який керує пристроєм, частиною якого є ця тактова кнопка, відбувається шляхом її натискання.

Коли користувач натискає тактову кнопку, відбувається замикання її контактів і від неї до мікроконтролера починає передаватися електричний сигнал, і він прямує до плати Arduino до того часу, поки контакти не будуть розімкнені.

З'єднання тактової кнопки і мікроконтролера повинно відбуватися з використанням резистора номіналом 10КΩ, оскільки якщо не використовувати його при їх з'єднанні, то виникає ефект брязкоту, який буде описано далі.

Тактові кнопки бувають наступних типів:

а) Тактова кнопка, що не фіксується, – тактова кнопка, яка сама повертається у вихідне положення після того, як користувач відпускає кнопку.

б) Тактова кнопка, що фіксується – тактова кнопка, яка змінює свій стан на протилежне тому, яке було до натискання і залишається в ньому після того, як користувач відпускає кнопку.

в) Тактова кнопка із залежною фіксацією – тактова кнопка, яка після того, як користувач натискає механічно пов'язану з нею іншу кнопку, автоматично переходить у положення «розтиснута».

А тепер про ефект брязкоту. Ефект брязкоту – це коли при роботі тактової кнопки виникають перешкоди, причиною яких є підключення тактової кнопки до плати Arduino без використання резистора.

Це призводить до того, що коли користувач натискає кнопку, всередині неї виникають мікроіскри, наслідком чого є те, що замість одного натискання

кнопки на мікроконтролер кожну мілісекунду передається по кілька її натискань, триває це доти, поки користувач її не відпустить. Ефект брязкоту є серйозною проблемою при роботі з тактовими кнопками, він може зробити прилад, який збирає розробник, неробочим. Тому використання резистора обов'язково.

Управління пристроєм «годинник-будильник» буде можливим двома способами: інфрачервоним пультом дистанційного керування або трьома тактовими кнопками, що не фіксуються.

Зовнішній вигляд тактової кнопки із чотирма пінами наведено на рис. 2.9.

Рисунок 2.9 – Тактова кнопка із чотирма пінами

2.6 Годинник реального часу DS1302 RTC

2.6.1 Модуль DS1302 (RTC): схема, опис

Годинник реального часу DS1302 є однією з моделей електронного годинникового механізму, який використовується при складанні різних пристроїв та механізмів, які повинні мати функціонал визначення, відліку та обліку актуального часу, тобто як правило в різних годинниках, які можуть бути як частиною складнішого механізму (наприклад, духова шафа, в яку вони можуть бути вбудовані), так і бути окремим повноцінним пристроєм (наприклад, це можуть бути настінні або наручні годинники).

У пристрої годинникового механізму є 5 виведень, які мають наступний функціонал:

VCC (колектор напруги) і GND (земля) – забезпечують живлення пристрою від зовнішнього джерела, ці висновки з'єднані з відповідними висновками плати Arduino Uno R3 CH340 на які живлення подається з плати, а на плату з якогось зовнішнього джерела, наприклад через USB-кабель від комп'ютера або через спеціальний акумуляторний відсік від батарейок або акумулятора.

CLK – це тактовий вивід годинника, який з'єднується з якимось із звичайних виведень плати Arduino Uno R3 CH340 і забезпечує правильну частоту роботи годинника.

DAT - це вивід передачі даних, через який здійснюється передача даних або з годинника на плату, або з плати на годинник. Напрямок передачі даних залежить від того, яку команду виконує годинник DS1302: команду передачі даних з плати на годинник (setDateTime) або з годинника на плату (getDateTime). При підключенні годинника DS1302 до плати Arduino Uno R3 CH340 цей вивід годинника з'єднується з будь-яким звичайним виведенням цієї плати.

RST - це вивід скидання даних на годиннику DS1302, використовується перед тим як встановити на годинник інший час, якщо до цього на них вже був встановлений час, цей вивід з'єднується з будь-яким звичайним виводом плати Arduino Uno R3 CH340.

Роботу, поведінку та функціонал годинного пристрою DS1302 також, як і роботу будь-якого іншого механічно-електронного пристрою, визначає програма-драйвер, яку раніше було написано, а потім завантажено у плату Arduino Uno R3 CH340 розробником цього пристрою (годинник). Для написання драйвера, керуючого цим годинником, існує безліч різних спеціальних бібліотек, що забезпечують можливість управління цим годинником. Я буду використовувати при написанні програми роботи цього годинника, методи та властивості, які містить бібліотека Ds1302.h (яку буде описано у розділі. 2.6.2).

Зовнішній вигляд годиннику реального часу DS1302 наведено на рис. 2.10.

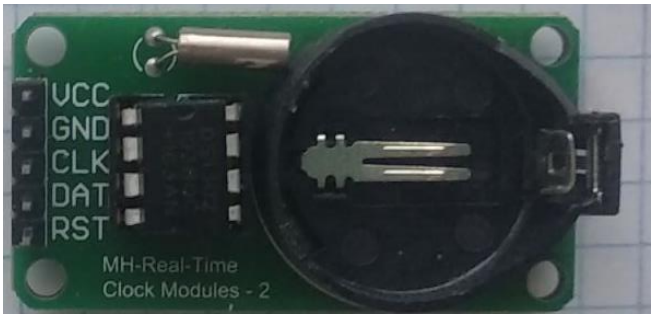

Рисунок 2.10 – Годинник реального часу DS1302

2.6.2 Бібліотека Ds1302.h Arduino: опис команд

Управління годинами реального часу відбувається за допомогою заздалегідь написаного коду, який визначає їх використання та поведінку, які прописані в цьому коді. Для того щоб написати такий код, необхідно використовувати в ньому спеціальні бібліотеки, які дозволяють працювати з годинником реального часу і керувати ними за допомогою методів і властивостей, що входять до цих бібліотек, що забезпечують роботу цих годин відповідно до програми користувача. У даній кваліфікаційній роботі я використовуватиму бібліотеку для роботи з годинником реального часу Ds1302.h методи та властивості якої описано нижче.

Підключення бібліотеки Ds1302.h має такий вигляд: `#include <Ds1302.h>`.

Ініціалізація бібліотеки має вигляд: `Ds1302 ідентифікатор_об'єкта_годин (номер_піна_RST, номер_піна_CLK, номер_піна_DAT).`

Для того, щоб ініціалізувати запис інформації в часовий механізм DS1302, треба використовувати метод `ідентифікатор_об'єкта_годин.init().`

Спеціально для роботи з датою, днем тижня та часом створюються дві спеціальні структури типу DateTime, синтаксис оголошення яких має такий

вигляд: `Ds1302::DateTime ідентифікатор_структури_дати_і_часу`. Одна з них створюється всередині методу setup і використовується для установки на годинник актуального часу, а друга з них створюється всередині методу loop і використовується для зчитування інформації з годинника реального часу DS1302 і для переведення годинника на інший час в деяких випадках, коли це необхідно (ці чотири випадки описані у постановці завдання).

Для встановлення на годинник реального часу актуального часу потрібно спочатку надати актуальні значення відповідним параметрам, що входять до першої структури, а потім передати дані з цієї структури на годинник реального часу. До структури DateTime входять такі параметри як:

а) `year` – параметр структури DateTime що містить у собі інформацію про рік.

б) `month` – параметр структури DateTime що містить у собі інформацію про місяць.

в) `day` – параметр структури DateTime що містить у собі інформацію про день.

г) `hour` – параметр структури DateTime що містить у собі інформацію про годину.

д) `minute` – параметр структури DateTime що містить у собі інформацію про хвилину.

е) `second` – параметр структури DateTime що містить у собі інформацію про секунду.

ж) `dow` - параметр структури DateTime що зберігає в собі інформацію про день тижня.

Для присвоєння значень параметрів структури DateTime відповідним параметрам годинника реального часу використовується спеціальний метод `ідентифікатор_об'єкта_годин.setDateTime(ідентифікатор_структури_дати_і_часу)`. При використанні цього методу дані із структури DateTime передаються на годинник реального часу та присвоюються відповідним параметрам об'єкта годинника реального часу.

Для запуску роботи годинника використовується метод `ідентифікатор_об'єкта_годин.halt()`.

Для того щоб отримати значення з годинника реального часу потрібно використовувати метод `ідентифікатор_об'єкта_годин.getDateTime (ідентифікатор_структури_дати_і_часу)`. При використанні цього методу, дані з годинника реального часу передаються відповідним параметрам структури DateTime і після цього значення параметрів можна зчитувати за тими ж назвами, за якими вони присвоювалися у випадку з першою структурою, але із вказівкою ідентифікатора другої структури. Також ця структура використовується для переведення годинника на інший час (у деяких випадках користувач повинен мати можливість перевести час з того, що перестало відповідати вимогам актуальності на правильний. Це може бути необхідно у таких випадках, як переведення годинника із зимового часу на літнє, з літнього часу на зимове, якщо годинник протягом деякого часу був позбавлений електричної енергії і за цей час відстав від актуального часу, і тому користувачеві треба перевести їх на актуальний час, а також, якщо користувач під час подорожі змінив часовий пояс, йому потрібно буде перевести годинник на ту кількість годин, яка відповідає тимчасовій різниці між двома часовими поясами). Переведення часу на годиннику відбувається таким чином:

а) За допомогою методу getDateTime передати дані з годинника реального часу об'єкту структури DateTime.

б) Після цього можна працювати з параметрами цього об'єкта DateTime, змінюючи їх на значення, які користувач хоче встановити як актуальний час.

в) Після того, як користувач вказав час, який він бажає встановити на годинник реального часу як актуальний час, потрібно передати ці значення часу на годинник за допомогою методу setDateTime, в результаті використання якого відповідним параметрам часу на годиннику будуть присвоєні обрані користувачем значення часу.

Тут наведено далеко не всі методи та властивості бібліотеки Ds1302.h, але тут перелічено всі властивості та методи цієї бібліотеки, які я використовуватиму у кваліфікаційній роботі.

2.7 Універсальні макетні плати для паяння

Для того, щоб збирати з великої кількості різних компонентів (датчики, кнопки, світлодіоди тощо) ті чи інші електронні пристрої та механізми, які розробник хоче зібрати, і які часто є складними, і складаються з великої кількості різних елементів, існують спеціальні макетні плати, до яких можна припаяти будь-які компоненти, що входять до схеми. При тому кількість компонентів, які можуть бути одночасно припаяні до такої плати, досить велика.

Цей елемент схеми є каркасом пристрою, до якого в міру збирання поетапно припаюються різні його елементи, наприклад, плата Arduino Uno, фоторезистор, чотиризначний 7-сегментний дисплей, потенціометр та багато інших пристроїв. У моїй дипломній роботі я припаюватиму до макетної плати такі елементи як:

а) Плата Arduino Uno R3 CH340 – 1 одиниця.

б) Приймач IR сигналу – 1 одиниця.

в) Активний зумер – 1 одиниця.

г) Чотирьохрозрядний 7-сегментний індикатор – 1 одиниця.

д) Тактова кнопка із чотирма пінами – 3 одиниці.

е) Годинник реального часу DS1302 RTC – 1 одиниця.

З'єднані ці елементи між собою проводами-перемичками макетної плати, один з кінців якого припаяний до виведення будь-якого з чотирьох останніх елементів списку, а інший до одного з виведень плати Arduino Uno R3 CH340. Ці дроти забезпечують можливість керувати різними пристроями, що входять до складу схеми годинника реального часу за допомогою плати Arduino Uno R3 CH340.

2.8 З'єднувальні дроти-перемички

Спеціально для з'єднання з першого боку будь-якого компонента пристрою, наприклад датчика, світлодіода, дисплея, кнопки, двигуна або іншого механізму, який розробник пристрою вирішить використовувати у своєму проекті, а з іншого – плати Arduino Uno, використовуються спеціальні з'єднувальні дроти, необхідні практично в будь-якому проекті створення будь-якого пристрою, оскільки через них проходить струм і електричний сигнал в обидві сторони: від мікроконтролера до компонента пристрою і від компонента до мікроконтролера, керуючого цим пристроєм. Без цих проводів було б неможливо передавати електричний струм між елементами схеми, отже, неможливо було б керувати пристроєм за допомогою мікроконтролера, а отже, було б неможливим і функціонування цього пристрою.

Проводи бувають різної довжини, типу і кольору, наприклад, основною відмінністю проводів різних типів є форма кінців цих проводів. Різні типи проводів необхідні тому, що для з'єднання різноманітних компонентів схеми потрібні проводи відповідних цим компонентам типів. У цьому проекті будуть використані дроти різних кольорів, довжин та типів, а використання дротів того чи іншого типу буде обумовлено лише конструктивною необхідністю.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Алгоритм налаштування часу дзвінка будильника та спосіб його реалізації.

Алгоритм налаштування часу дзвінка будильника виглядає так:

а) Для початку налаштування часу дзвінка будильника треба або натиснути кнопку, яка знаходиться між двома іншими тактовими кнопками і припаяна до друкованої плати, або натиснути кнопку «1» на інфрачервоному пульті дистанційного керування.

б) Для того, щоб додати 1 хвилину до часу, що налаштовується, треба натиснути праву тактову кнопку припаяну до друкованої плати або кнопку «+» на інфрачервоному пульті дистанційного керування, а щоб відняти 1 хвилину від настроюваного часу, треба натиснути ліву тактову кнопку пристрою або кнопку «-» на пульті.

в) Для того, щоб завершити налаштування часу дзвінка будильника, треба або натиснути кнопку, яка знаходиться між двома іншими тактовими кнопками і припаяна до друкованої плати або натиснути кнопку «4» на інфрачервоному пульті дистанційного керування.

Після того як користувач встановить час дзвінка будильника, користувачеві залишиться тільки дочекатися дзвінка будильника і вимкнути його одним з двох способів: або натиснувши кнопку, яка знаходиться між двома іншими тактовими кнопками і припаяна до друкованої плати, або натиснувши кнопку «3» на інфрачервоному пульті дистанційного керування.

Три етапи налаштування часу дзвінка будильника, які були описані вище, реалізуються у вигляді трьох вкладених один в одного шарів коду:

а) Реалізація першого шару коду, який забезпечує функціонал входу в меню налаштування часу дзвінка будильника, здійснюється за допомогою двох операторів «if» та одного нескінченного циклу «for», перший оператор «if» перевіряє, чи знаходиться центальна кнопка в натиснутому стані чи ні. В нього вкладений перший нескінченний цикл «for», який у свою чергу забезпечує очікування того моменту, коли користувач відпустить центральну кнопку «входу/виходу» в меню налаштування будильника. В перший нескінченний цикл «for» у свою чергу вкладений другий оператор «if», який використовується для перевірки того чи була відпущена центральна кнопка, чи ні.

б) Реалізація другого шару коду, який забезпечує функціонал налаштування часу, коли повинен задзвеніти будильник, здійснюється за допомогою вкладеного в другий оператор «if» другого нескінченного циклу «for», в якому відбувається налаштування часу дзвінка будильника. Для неї

використовуються дві змінні типу «byte», одна з яких містить годину в який повинен задзвеніти будильник, а друга – хвилину в яку будильник повинен почати дзвеніти.

в) Реалізація третього шару коду, який забезпечує функціонал виходу з меню налаштування часу дзвінка будильника, здійснюється за допомогою двох операторів «if» та одного нескінченного циклу «for». У другий нескінченний цикл «for» вкладений третій оператор «if», який перевіряє, чи була натиснута вдруге кнопка «входу/виходу» з меню налаштування будильника. У третій оператор «if» вкладено третій нескінченний цикл «for», який забезпечує очікування того моменту, коли користувач відпустить вдруге натиснуту кнопку «входу/виходу» з меню налаштування будильника. В цей цикл у свою чергу вкладено четвертий оператор «if», який забезпечує виконання перевірки, чи була відпущена середня кнопка або ні після другого натискання. Коли програма заходить всередину цього оператора, відбувається почерговий вихід з усіх трьох нескінченних циклів «for».

Алгоритм установки часу дзвінку будильника має наступний вигляд.

```
if(alarm_time_setting_status == HIGH)
//Якщо користувач натиснув кнопку початку настройки дзвінка будильника.
      {
        byte hour_of_the_alarm = timing_structure.hour;
//Передаємо дані години зі структури timing_structure на змінну hour_of_the_alarm.
        byte minute_of_the_alarm = timing_structure.minute;
//Передаємо дані хвилини із структури timing_structure на змінну minute_of_the_alarm.
   for(;;)
//Запускаємо нескінченний цикл.
        {
          displaying_a_number(hour_of_the_alarm,
minute_of_the_alarm);    //Виводимо обрані дані на дисплей.
```

```
          alarm_time_setting_status                                    =
digitalRead(second_tact_button);                  //Зчитуємо дані з
кнопки початку настройки часу дзвінка будильника.

          if(alarm_time_setting_status          ==          LOW)
//Якщо користувач натиснув кнопку початку настройки часу дзвінка
будильника.

          {

            for(;;)
//Запускаємо нескінченний цикл.

            {

              displaying_a_number(hour_of_the_alarm,
minute_of_the_alarm);   //Виводимо обрані дані на дисплей.

              byte           decrease_the_alarm_time           =
digitalRead(first_tact_button);         //Зчитуємо дані з кнопки
зменшення часу дзвінка будильника.

              alarm_time_setting_status                        =
digitalRead(second_tact_button);                //Зчитуємо дані з
кнопки початку-кінця настройки будильника.

              byte           increase_the_alarm_time           =
digitalRead(third_tact_button);         //Зчитуємо дані з кнопки
збільшенні часу дзвінка будильника.

              if(decrease_the_alarm_time         ==         HIGH)
//Якщо користувач натиснув кнопку зменшення часу дзвінка
будильника.

              {

                for(;;)
//Запускаємо нескінченний цикл.

                {

                  displaying_a_number(hour_of_the_alarm,
minute_of_the_alarm);   //Виводимо обрані дні на дисплей.

                  decrease_the_alarm_time                      =
digitalRead(first_tact_button);                   //Зчитуємо дані
з кнопки зменшення часу дзвінка будильника.

                  if(decrease_the_alarm_time       ==       LOW)
//Якщо користувач натискає кнопку зменшення часу дзвінка
будильника.

                  {
```

```
                    if(minute_of_the_alarm          >          0)
//Якщо хвилина дзвінка будильника більше 0.
                    {
                      minute_of_the_alarm          -=          1;
//Віднімаємо одну хвилину.
                    }
                    else     if(minute_of_the_alarm     ==     0)
//Якщо хвилина дзвінка будильника дорівнює 0.
                    {
                      minute_of_the_alarm         +=         59;
//Додаємо 59 мінут.
                      if(hour_of_the_alarm          >          0)
//Якщо час дзвінка будильника більше 0.
                      {
                        hour_of_the_alarm          -=          1;
//Віднімаємо одну годину.
                      }
                      else     if(hour_of_the_alarm     ==     0)
//Якщо година дзвінка будильника дорівнює 0.
                      {
                        hour_of_the_alarm          =          23;
//Дорівнюємо годину дзвінка будильника 23.
                      }
                    }
                    break;
//Завершуємо нескінченний цикл.
                  }
                }
              }
              else    if    (increase_the_alarm_time    ==    HIGH)
//Якщо користувач натиснув кнопку додавання часу.
              {
                for(;;)
//Запускаємо нескінченний цикл.
```

```
                {
                  displaying_a_number(hour_of_the_alarm,
minute_of_the_alarm);   //Виводимо обрані данні на дисплей.
                  increase_the_alarm_time                        =
digitalRead(third_tact_button);                //Зчитуємо данні з
кнопки додавання часу.
                  if     (increase_the_alarm_time     ==     LOW)
//Якщо користувач натиснув кнопку додавання часу.
                  {
                    if(minute_of_the_alarm          <          59)
//Якщо мінута дзвінка будильника менше 59.
                    {
                      minute_of_the_alarm          +=          1;
//Додаємо 1 хвилину.
                    }
                    else     if(minute_of_the_alarm     ==     59)
//Якщо хвилина дзвінка будильника дорівнює 59.
                    {
                      minute_of_the_alarm          -=          59;
//Віднімаємо 59 мінут.
                      if(hour_of_the_alarm          <          23)
//Якщо час дзвінка годинника менше 23.
                      {
                        hour_of_the_alarm          +=          1;
//Додаємо одну годину.
                      }
                      else     if(hour_of_the_alarm     ==     23)
//Якщо час дзвінка будильника дорівнює 0.
                      {
                        hour_of_the_alarm          =          0;
//Дорівнюємо годину дзвінка будильника 0.
                      }
                    }
                    break;
//Завершуємо нескінченний цикл.
```

```
                }
              }
            }
            if(alarm_time_setting_status        ==        HIGH)
// Якщо користувач натиснув кнопку настройки будильника.
            {
              for(;;)
//Запускаємо нескінченний цикл.
              {
                displaying_a_number(hour_of_the_alarm,
minute_of_the_alarm);   //Виводимо обрані дані на дисплей.
                alarm_time_setting_status                    =
digitalRead(second_tact_button);              //Зчитуємо дані з
кнопки початку-закінчення настройки будильника.
                if(alarm_time_setting_status      ==      LOW)
//Якщо користувач відпустив кнопка настройки будильника.
                {
                  break;
//Завершуємо нескінченний цикл.
                }
              }
              break;
//Завершуємо нескінченний цикл.
            }
          }
          break;
//Завершуємо нескінченний цикл.
        }
      }
    }
```

3.2 Алгоритм переведення часу встановленого на годиннику та спосіб його реалізації

Алгоритм переведення часу, встановленого на годиннику, виглядає так:

а) Для того щоб почати процес переведення часу, встановленого на годиннику, треба або натиснути кнопку, яка знаходиться зліва від двох інших тактових кнопок, і так само як і вони, припаяна до друкованої плати, або натиснути кнопку «2» на інфрачервоному пульті дистанційного керування.

б) Для того щоб додати 1 хвилину до настроюваного часу треба або натиснути праву тактову кнопку пристрою, або натиснути кнопку «+» на інфрачервоному пульті дистанційного керування. Щоб відняти 1 хвилину від настроюваного часу, треба натиснути або кнопку, яка знаходиться між двома іншими тактовими кнопками та припаяна до друкованої плати, або кнопку «-» на пульті.

в) Для того щоб встановити обраний час на годинник реального часу DS1302, треба або натиснути кнопку, яка знаходиться зліва від двох інших тактових кнопок, і так само як і вони, припаяні до друкованої плати, або натиснути кнопку «5» на інфрачервоному пульті дистанційного керування.

Після виконання цього алгоритму дій на годиннику стоятиме вже не той час, який стояв до цього, а новий, який вказав користувач, і який був встановлений на них відразу ж після виходу з меню налаштування часу.

Три етапи перекладу часу, встановленого на годиннику, який був описаний вище, реалізуються у вигляді трьох вкладених один в одного шарів коду:

а) Реалізація першого шару коду, який забезпечує функціонал входу в меню переведення часу встановленого на годиннику, здійснюється за допомогою двох операторів «if» та одного нескінченного циклу «for», перший оператор «if» перевіряє чи знаходиться ліва кнопка в натиснутому стані чи ні, в нього вкладений перший нескінченний цикл «for» , який у свою чергу забезпечує очікування того моменту, коли користувач відпустить центральну кнопку «входу/виходу» в меню переведення часу встановленого на годиннику. В перший нескінченний цикл «for» у свою чергу вкладений

другий оператор «if», який використовується для перевірки того, чи була відпущена ліва кнопка чи ні.

б) Реалізація другого шару коду, який забезпечує функціонал переказу часу, встановленого на годиннику, здійснюється за допомогою вкладеного в другий оператор «if» другого нескінченного циклу «for», в якому відбувається вибір часу, який користувач хоче встановити на годиннику. В ньому використовуються дві змінні типу «byte», одна з яких містить годину, яку користувач хоче встановити на годинник, а друга – хвилину, яку користувач хоче встановити на годинник.

в) Реалізація третього шару коду, який забезпечує функціонал виходу з меню переведення часу, встановленого на годиннику, здійснюється за допомогою двох операторів «if» та одного нескінченного циклу «for», у другий нескінченний цикл «for» вкладений третій оператор «if», який перевіряє чи була натиснута вдруге кнопка «входу/виходу» з меню переведення часу встановленого на годиннику. В третій оператор «if» вкладено третій нескінченний цикл «for», який забезпечує очікування того моменту, коли користувач відпустить вдруге натиснуту кнопку «входу/виходу» з меню переведення часу, встановленого на годиннику. У цей цикл у свою чергу вкладено четвертий оператор «if», який забезпечує виконання перевірки того, чи відпущена ліва кнопка після другого натискання чи ні. Коли програма заходить всередину цього оператора, відбувається почерговий вихід з усіх трьох нескінченних циклів «for».

Алгоритм переведення часу, втсановленого на годиннику, має наступний вигляд:

```
else if(clock_time_change_status == HIGH)
//Якщо користувач натиснув кнопку переведення часу годинника.
      {
        byte hour_standing_on_the_clock = timing_structure.hour;
// Присвоюємо змінній години значення зі структури
timing_structure.
```

```
        byte minute_standing_on_the_clock =
timing_structure.minute;
//Присвоюємо змінній хвилині значення зі структури
timing_structure.

        for(;;)
//Запускаємо нескінченний цикл.

        {

          displaying_a_number(hour_standing_on_the_clock,
minute_standing_on_the_clock);  //Виводимо обрані дані на
дисплей.

          clock_time_change_status =
digitalRead(first_tact_button);
//Зчитуємо дані з кнопки переведення часу.

          if(clock_time_change_status == LOW)
//Якщо користувач відпустив кнопку переведення часу годинника.

          {

            for(;;)
//Запускаємо нескінченний цикл.

            {

              displaying_a_number(hour_standing_on_the_clock,
minute_standing_on_the_clock);   //Виводимо обрані дані на
дисплей.

              byte decrease_clock_time =
digitalRead(second_tact_button);
//Зчитуємо дані з кнопки віднімання часу.

              clock_time_change_status =
digitalRead(first_tact_button);
//Зчитуємо дані з кнопки переведення часу.

              byte increase_clock_time =
digitalRead(third_tact_button);
//Зчитуємо дані з кнопки додавання часу.

              if(decrease_clock_time == HIGH)
//Якщо користувач натиснув кнопку віднімання часу.

              {

                for(;;)
//Запускаємо нескінченний цикл.

                {
```

```
displaying_a_number(hour_standing_on_the_clock,
minute_standing_on_the_clock);   //Виводимо обрані дані на
дисплей.

                  decrease_clock_time =
digitalRead(second_tact_button);
//Зчитуємо данні з кнопки віднімання часу.

                  if(decrease_clock_time == LOW)
//Якщо користувач відпустив кнопку віднімання часу.

                  {

                    if(minute_standing_on_the_clock > 0)
//Якщо хвилина більше 0.

                    {

                      minute_standing_on_the_clock -= 1;
//Зменшуємо хвилину на 1.

                    }

                    else if(minute_standing_on_the_clock == 0)
//Якщо кількість хвилин дорівнює 0.

                    {

                      minute_standing_on_the_clock += 59;
//Додаємо 59 мінут.

                      if(hour_standing_on_the_clock > 0)
//Якщо година більше 0.

                      {

                        hour_standing_on_the_clock -= 1;
//Зменшуємо годину на 1.

                      }

                      else if(hour_standing_on_the_clock == 0)
//Якщо година дорівнює 0.

                      {

                        hour_standing_on_the_clock = 23;
//Дорівнюємо годину 23.

                      }

                    }

                    break;
//Завершуємо нескінченний цикл.
```

```
                }
              }
            }
            else if(increase_clock_time == HIGH)
//Якщо користувач натиснув кнопку додавання часу годинника.
            {
              for(;;)
//Запускаємо нескінченний цикл.
              {

displaying_a_number(hour_standing_on_the_clock,
minute_standing_on_the_clock);  //Виводимо обрані значення на
дисплей.
                increase_clock_time =
digitalRead(third_tact_button);
//Зчитуємо дані з кнопки додавання часу.
                if(increase_clock_time == LOW)
//Якщо користувач відпустив кнопку додавання часу.
                {
                  if(minute_standing_on_the_clock < 59)
//Якщо хвилина менше ніж 59.
                  {
                    minute_standing_on_the_clock += 1;
//Додаємо 1 хвилину.
                  }
                  else if(minute_standing_on_the_clock == 59)
//Якщо хвилина дорівнює 59.
                  {
                    minute_standing_on_the_clock -= 59;
//Віднімаємо 59 мінут.
                    if(hour_standing_on_the_clock < 23)
//Якщо час дорівнює менше 23 годин.
                    {
                      hour_standing_on_the_clock += 1;
//Додаємоодну годину.
```

```
                    }
                    else if(hour_standing_on_the_clock == 23)
//Якщо час дорівнює 23 годинам.
                    {
                      hour_standing_on_the_clock = 0;
//Присвоюємо годині часу значення 0.
                    }
                  }
                  break;
//Завершуємо нескінченний цикл.
                }
              }
            }
            if(clock_time_change_status == HIGH)
//Якщо користувач другий раз натиснув кнопку переведення часу годинника.
            {
              for(;;)
//Запускаємо нескінченний цикл.
              {

displaying_a_number(hour_standing_on_the_clock,
minute_standing_on_the_clock);  //Виводимо обрані значення на дисплей.
                clock_time_change_status =
digitalRead(first_tact_button);
//Зчитуємо дані з кнопки переведення часу.
                if(clock_time_change_status == LOW)
//Якщо користувач другий раз відпустив кнопку переведення часу годинника.
                {
                  break;
//Завершуємо нескінченний цикл.
                }
              }
```

```
                break;
//Завершуємо нескінченний цикл.
              }
            }
            break;
//Завершуємо нескінченний цикл.
          }
        }
        timing_structure.hour = hour_standing_on_the_clock;
//Передаємо обрану користувачем годину структурі.
        timing_structure.minute = minute_standing_on_the_clock;
//Передаємо обрану користувачем мінуту структурі.
        real_time_clock_object.setDateTime(&timing_structure);
//Передаємо данні зі структури timing_structure на годинник
реального часу.
      }
```

4 ПРАКТИЧНА ЧАСТИНА

4.1 Результати виконання дипломної роботи

У драйвері цього пристрою була розроблена система методів, призначених для виведення різних цифер на різні розряди, ці висновки мають таку структуру:

```
void number_0_on_the_first_display ()      //Метод виводу цифри
«0» на перший розряд числа.
{
     digitalWrite(D1, HIGH);                          //
Виведення першого розряду числа.
```

```
      digitalWrite(D2, LOW);                          /Виведення
другого розряду числа.
      digitalWrite(D3, LOW);                          // Виведення
третього розряду числа.
      digitalWrite(D4, LOW);                          // Виведення
дчетвертого розряду числа.
      digitalWrite(a, LOW);                           //Виведення
верхнього світлодіоду.
      digitalWrite(b, LOW);                           //Виведення
верхнього правого світлодіоду.
      digitalWrite(c, LOW);                           // Виведення
нижнього правого світлодіоду.
      digitalWrite(d, LOW);                           //Виведення
нижнього світлодіоду.
      digitalWrite(e, LOW);                           // Виведення
нижнього лівого світлодіоду.
      digitalWrite(f, LOW);                           // Виведення
верхнього лівого світлодіоду.
      digitalWrite(g, HIGH);                          // Виведення
середнього світлодіоду.
}
```

Кількість таких методів дорівнює:

Для першого розряду – 3 штуки (цифри 0, 1, 2).

Для другого розряду – 10 шт (цифри 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Для третього розряду – 6 штук (цифри 0, 1, 2, 3, 4, 5).

Для четвертого розряду – 10 шт (цифри 0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

Спеціальний метод сбросу виведень `zeroing_of_conclusions ()`.

Усього цих методів 3+10+6+10+1=30.

Ці 30 методів виведення значень часу на чотиризначний 7-сегментний світлодіодний дисплей викликається через спеціальний метод у задачу якого входить забезпечення найбільш зручного способу їх використання при виведенні часу на дисплей моєї моделі електронного годиннику.

Синтаксис опису цього методу виглядає наступним чином:

```
void displaying_a_number(byte hour_of_time, byte minute_of_time)
// Метод виведення часу на чотиризначний 7-сегментний
світлодіодний дисплей.
{
  if(hour_of_time / 10 == 0)                  // Якщо у першому
розряді має бути 0.
  {
    number_0_on_the_first_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time / 10 == 1)          // Якщо у першому
розряді має бути 1.
  {
    number_1_on_the_first_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time / 10 == 2)          // Якщо у першому
розряді має бути 2.
  {
    number_2_on_the_first_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  zeroing_of_conclusions ();                   // Скидаємо
висновки.
  if(hour_of_time % 10 == 0)                 // Якщо у другому
розряді має бути 0.
  {
    number_0_on_the_second_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 1)              // Якщо у другому
розряді має бути 1.
  {
    number_1_on_the_second_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
```

```
  }
  else if(hour_of_time % 10 == 2)             // Якщо у другому
розряді має бути 2.
  {
    number_2_on_the_second_display ();//Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 3)             // Якщо у другому
розряді має бути 3.
  {
    number_3_on_the_second_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 4)              // Якщо у другому
розряді має бути 4.
  {
    number_4_on_the_second_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 5)               // Якщо у
другому розряді має бути 5.
  {
    number_5_on_the_second_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 6)               // Якщо у
другому розряді має бути 6.
  {
    number_6_on_the_second_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 7)               // Якщо у
другому розряді має бути 7.
  {
```

```
    number_7_on_the_second_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 8)                // Якщо у
другому розряді має бути 8.
  {
    number_8_on_the_second_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(hour_of_time % 10 == 9)               // Якщо у другому
розряді має бути 9.
  {
    number_9_on_the_second_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  zeroing_of_conclusions ();                          // Скидаємо
висновки.
  if(minute_of_time / 10 == 0)                      // Якщо у
третьому розряді має бути 0.
  {
    number_0_on_the_third_display ();     // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time / 10 == 1)             // Якщо у
третьому розряді має бути 1.
  {
    number_1_on_the_third_display ();    // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time / 10 == 2)    // Якщо у третьому
розряді має бути 2.
  {
    number_2_on_the_third_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
```

```
  else if(minute_of_time / 10 == 3)        // Якщо у третьому
розряді має бути 3.
  {
    number_3_on_the_third_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time / 10 == 4)        // Якщо у третьому
розряді має бути 4.
  {
    number_4_on_the_third_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time / 10 == 5)         // Якщо у третьому
розряді має бути 5.
  {
    number_5_on_the_third_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  zeroing_of_conclusions ();                    // Скидаємо
висновки.
  if(minute_of_time % 10 == 0)              // Якщо у четвертому
розряді має бути 0.
  {
    number_0_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 1)         // Якщо у четвертому
розряді має бути 1.
  {
    number_1_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 2)         // Якщо у четвертому
розряді має бути 2.
  {
```

```
    number_2_on_the_fourth_display (); // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 3)        // Якщо у четвертому
розряді має бути 3.
  {
    number_3_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 4)           // Якщо у
четвертому розряді має бути 4.
  {
    number_4_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 5)          // Якщо у
четвертому розряді має бути 5.
  {
    number_5_on_the_fourth_display ();   // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 6)         // Якщо у четвертому
розряді має бути 6.
  {
    number_6_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 7)          // Якщо у
четвертому розряді має бути 7.
  {
    number_7_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 8)         // Якщо у четвертому
розряді має бути 8.
```

```
  {
    number_8_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  else if(minute_of_time % 10 == 9)          // Якщо у четвертому
розряді має бути 9.
  {
    number_9_on_the_fourth_display ();  // Використовуємо
відповідний спосіб виведення цифри на розряд дисплея.
  }
  zeroing_of_conclusions ();                          // Скидаємо
висновки.
}
```

Вище описані методи дозволяють працювати з чотиризначним 7-сегменним світлодіодним дисплеєм та керувати ним найбільш зручним чином.

Коди, розроблені для програмування роботи пристрою наведено у ДОДАТКУ А.

ВИСНОВКИ

Дана дипломна робота спрямована на проектування та складання електронного пристрою «годинник-будильник», який може використовуватися двома способами: як годинник і як будильник, при цьому обидва ці завдання можуть виконуватися одночасно. Складання цього пристрою було здійснено на основі мікроконтролера Arduino Uno, який керує пристроєм. Також при його складанні були використані й інші електронні елементи, такі як інфрачервоний пульт та інфрачервоний приймач дистанційного керування, активний зумер, чотирирозрядний 7-сегментний світлодіодний індикатор, три тактові кнопки з чотирма пінами кожна та годинник реального часу DS1302. З усіх вищеперелічених частин пристрою було зібрано один цілий пристрій годинника-будильника, який визначає та враховує час, виводить на чотиризначний 7-сегментний світлодіодний дисплей або актуальний час, або час дзвінка будильника, який користувач налаштовує, або час який користувач хоче встановити на годинник замість попереднього. Також цей пристрій дзвонить в заздалегідь вказаний користувачем час і має два функціонали налаштування: функціонал переведення годинника на інший час і функціонал налаштування часу дзвінка будильника, при цьому будь-яким з цих функціоналів можна скористатися двома способами: або за допомогою тактових кнопок, або використовуючи пульт дистанційного управління.

У роботі описані різні елементи пристрою з яких було зібрано пристрій годинник-будильник, а також необхідне для написання драйвера до цього пристрою програмне забезпечення.

Розглянуто особливості створення електронних пристроїв на основі мікроконтролера Arduino Uno R3 CH340 та вивчено способи керування цими пристроями.

Проведене мною проектування, розробка та складання пристрою завершилися успішно, було створено пристрій, який успішно виконує поставлені перед ним завдання та працює без помилок.

Зібраний пристрій має перспективи подальшого розвитку, в тому числі подальше опрацювання функцій налаштування часу, які можна опрацювати більш детально і покращити їх алгоритми, наприклад, зробивши два варіанти функції налаштування часу будильника: вибір часу для одноразового дзвінка будильника і вибір часу для багаторазового дзвінка будильника.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Годинник [Електроний ремурс] / Україна – Режим доступу : \www/ URL: https://uk.wikipedia.org/wiki/%D0%93%D0%BE%D0%B4%D0%B8%D0%BD%D0%BD%D0%B8%D0%BA – 11.08.2022 г – Загл.з екрану.
2. HUM: REMOTE CONTROLLER + IR RECEIVER [Електроний ресурс] / UK – Режим доступу : \www/ URL: https://soldered.com/learn/hum-remote-controller-ir-receiver/

ДОДАТОК А

```
#include <Ds1302.h>
byte buzzer = 7;
byte D1 = A5;
byte D2 = A2;
byte D3 = A1;
byte D4 = 5;
byte a = A4;
byte b = A0;
byte c = 3;
byte d = 2;
byte e = 6;
byte f = A3;
byte g = 4;
byte first_tact_button = 10;
byte second_tact_button = 9;
byte third_tact_button = 8;
Ds1302 real_time_clock_object(13, 11, 12);
void number_0_on_the_first_display ()
{
      digitalWrite(D1, HIGH);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
```

```
      digitalWrite(g, HIGH);
}
void number_1_on_the_first_display ()
{
      digitalWrite(D1, HIGH);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_2_on_the_first_display ()
{
      digitalWrite(D1, HIGH);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_0_on_the_second_display ()
{
```

```
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, HIGH);
}
void number_1_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_2_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
```

```
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_3_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_4_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
```

```
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_5_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_6_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
```

```
void number_7_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_8_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_9_on_the_second_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, HIGH);
```

```
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_0_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, HIGH);
}
void number_1_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
```

```
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_2_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_3_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
```

```
      digitalWrite(g, LOW);
}
void number_4_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_5_on_the_third_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, HIGH);
      digitalWrite(D4, LOW);
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_0_on_the_fourth_display ()
{
```

```
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, HIGH);
}
void number_1_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_2_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
```

```
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_3_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, LOW);
}
void number_4_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
```

```
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_5_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_6_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
```

```
void number_7_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
}
void number_8_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, LOW);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void number_9_on_the_fourth_display ()
{
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
```

```
      digitalWrite(D3, LOW);
      digitalWrite(D4, HIGH);
      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      digitalWrite(e, HIGH);
      digitalWrite(f, LOW);
      digitalWrite(g, LOW);
}
void zeroing_of_conclusions ()
{
      digitalWrite(a, HIGH);
      digitalWrite(b, HIGH);
      digitalWrite(c, HIGH);
      digitalWrite(d, HIGH);
      digitalWrite(e, HIGH);
      digitalWrite(f, HIGH);
      digitalWrite(g, HIGH);
      digitalWrite(D1, LOW);
      digitalWrite(D2, LOW);
      digitalWrite(D3, LOW);
      digitalWrite(D4, LOW);
}
void setup() {
      Serial.begin(9600);
      pinMode(buzzer, OUTPUT);
      pinMode(D1, OUTPUT);
      pinMode(D2, OUTPUT);
      pinMode(D3, OUTPUT);
      pinMode(D4, OUTPUT);
      pinMode(a, OUTPUT);
```

```
      pinMode(b, OUTPUT);
      pinMode(c, OUTPUT);
      pinMode(d, OUTPUT);
      pinMode(e, OUTPUT);
      pinMode(f, OUTPUT);
      pinMode(g, OUTPUT);
      pinMode(first_tact_button, INPUT);
      pinMode(second_tact_button, INPUT);
      pinMode(third_tact_button, INPUT);
      real_time_clock_object.init();
      Ds1302::DateTime start_time_structure;
      start_time_structure.year = 2023;
      start_time_structure.month = 6;
      start_time_structure.day = 18;
      start_time_structure.hour = 16;
      start_time_structure.minute = 10;
      start_time_structure.second = 10;
      start_time_structure.dow = 7;
      real_time_clock_object.setDateTime(&start_time_structure);
      real_time_clock_object.halt();
}
void loop() {
      byte alarm_time_setting_status =
digitalRead(second_tact_button);
      byte clock_time_change_status =
digitalRead(first_tact_button);
      Ds1302::DateTime timing_structure;
      real_time_clock_object.getDateTime(&timing_structure);
      if(alarm_time_setting_status == HIGH)
      {
        byte hour_of_the_alarm = timing_structure.hour;
        byte minute_of_the_alarm = timing_structure.minute;
```

```
        for(;;)
        {
          number_0_on_the_fourth_display ();
          alarm_time_setting_status =
digitalRead(second_tact_button);
          if(alarm_time_setting_status == LOW)
          {
            for(;;)
            {
              number_2_on_the_fourth_display ();
              byte decrease_the_alarm_time =
digitalRead(first_tact_button);
              alarm_time_setting_status =
digitalRead(second_tact_button);
              byte increase_the_alarm_time =
digitalRead(third_tact_button);
              if(decrease_the_alarm_time == HIGH)
              {
                for(;;)
                {
                  number_6_on_the_fourth_display ();
                  decrease_the_alarm_time =
digitalRead(first_tact_button);
                  if(decrease_the_alarm_time == LOW)
                  {
                    if(minute_of_the_alarm > 0)
                    {
                      Serial.print(hour_of_the_alarm);
                      Serial.print(":");
                      Serial.println(minute_of_the_alarm);
                      Serial.println();
                      minute_of_the_alarm -= 1;
                      Serial.print(hour_of_the_alarm);
                      Serial.print(":");
```

```
        Serial.println(minute_of_the_alarm);
        Serial.println();
      }
      else if(minute_of_the_alarm == 0)
      {
        Serial.print(hour_of_the_alarm);
        Serial.print(":");
        Serial.println(minute_of_the_alarm);
        Serial.println();
        minute_of_the_alarm += 59;
        if(hour_of_the_alarm > 0)
        {
          hour_of_the_alarm -= 1;
        }
        else if(hour_of_the_alarm == 0)
        {
          hour_of_the_alarm += 23;
        }
        Serial.print(hour_of_the_alarm);
        Serial.print(":");
        Serial.println(minute_of_the_alarm);
        Serial.println();
      }
      break;
    }
  }
}
else if (increase_the_alarm_time == HIGH)
{
  for(;;)
  {
    number_8_on_the_fourth_display ();
```

```
                    increase_the_alarm_time =
digitalRead(third_tact_button);
                    if (increase_the_alarm_time == LOW)
                    {
                      if(minute_of_the_alarm < 59)
                      {
                        Serial.print(hour_of_the_alarm);
                        Serial.print(":");
                        Serial.println(minute_of_the_alarm);
                        Serial.println();
                        minute_of_the_alarm += 1;
                        Serial.print(hour_of_the_alarm);
                        Serial.print(":");
                        Serial.println(minute_of_the_alarm);
                        Serial.println();
                      }
                      else if(minute_of_the_alarm == 59)
                      {
                        Serial.print(hour_of_the_alarm);
                        Serial.print(":");
                        Serial.println(minute_of_the_alarm);
                        Serial.println();
                        minute_of_the_alarm -= 59;
                        if(hour_of_the_alarm < 23)
                        {
                          hour_of_the_alarm += 1;
                        }
                        else if(hour_of_the_alarm == 23)
                        {
                          hour_of_the_alarm = 0;
                        }
                        Serial.print(hour_of_the_alarm);
```

```
                    Serial.print(":");
                    Serial.println(minute_of_the_alarm);
                    Serial.println();
                  }
                  break;
                }
              }
            }
            if(alarm_time_setting_status == HIGH)
            {
              for(;;)
              {
                number_4_on_the_fourth_display ();
                alarm_time_setting_status =
digitalRead(second_tact_button);
                if(alarm_time_setting_status == LOW)
                {
                  break;
                }
              }
              break;
            }
          }
          break;
        }
      }
      else if(clock_time_change_status == HIGH)
      {
        byte hour_standing_on_the_clock = timing_structure.hour;
        byte minute_standing_on_the_clock =
timing_structure.minute;
```

```
        for(;;)
        {
          number_1_on_the_fourth_display ();
          clock_time_change_status =
digitalRead(first_tact_button);
          if(clock_time_change_status == LOW)
          {
            for(;;)
            {
              number_3_on_the_fourth_display ();
              byte decrease_clock_time =
digitalRead(second_tact_button);
              clock_time_change_status =
digitalRead(first_tact_button);
              byte increase_clock_time =
digitalRead(third_tact_button);
              if(decrease_clock_time == HIGH)
              {
                for(;;)
                {
                  number_7_on_the_fourth_display ();
                  decrease_clock_time =
digitalRead(second_tact_button);
                  if(decrease_clock_time == LOW)
                  {
                    if(minute_standing_on_the_clock > 0)
                    {
                      Serial.print(hour_standing_on_the_clock);
                      Serial.print(":");

Serial.println(minute_standing_on_the_clock);
                      Serial.println();
                      minute_standing_on_the_clock -= 1;
                      Serial.print(hour_standing_on_the_clock);
```

```
                    Serial.print(":");
Serial.println(minute_standing_on_the_clock);
                    Serial.println();
                  }
                  else if(minute_standing_on_the_clock == 0)
                  {
                    Serial.print(hour_standing_on_the_clock);
                    Serial.print(":");
Serial.println(minute_standing_on_the_clock);
                    Serial.println();
                    minute_standing_on_the_clock += 59;
                    if(hour_standing_on_the_clock > 0)
                    {
                      hour_standing_on_the_clock -= 1;
                    }
                    else if(hour_standing_on_the_clock == 0)
                    {
                      hour_standing_on_the_clock += 23;
                    }
                    Serial.print(hour_standing_on_the_clock);
                    Serial.print(":");
Serial.println(minute_standing_on_the_clock);
                    Serial.println();
                  }
                  break;
                }
              }
            }
            else if(increase_clock_time == HIGH)
            {
```

```
                for(;;)
                {
                  number_9_on_the_fourth_display ();
                  increase_clock_time =
digitalRead(third_tact_button);
                  if(increase_clock_time == LOW)
                  {
                    if(minute_standing_on_the_clock < 59)
                    {
                      Serial.print(hour_standing_on_the_clock);
                      Serial.print(":");

Serial.println(minute_standing_on_the_clock);
                      Serial.println();
                      minute_standing_on_the_clock += 1;
                      Serial.print(hour_standing_on_the_clock);
                      Serial.print(":");

Serial.println(minute_standing_on_the_clock);
                      Serial.println();
                    }
                    else if(minute_standing_on_the_clock == 59)
                    {
                      Serial.print(hour_standing_on_the_clock);
                      Serial.print(":");

Serial.println(minute_standing_on_the_clock);
                      Serial.println();
                      minute_standing_on_the_clock -= 59;
                      if(hour_standing_on_the_clock < 23)
                      {
                        hour_standing_on_the_clock += 1;
                      }
```

```
                    else if(hour_standing_on_the_clock == 23)
                    {
                      hour_standing_on_the_clock = 0;
                    }
                    Serial.print(hour_standing_on_the_clock);
                    Serial.print(":");

Serial.println(minute_standing_on_the_clock);
                    Serial.println();
                  }
                  break;
                }
              }
            }
            if(clock_time_change_status == HIGH)
            {
              for(;;)
              {
                number_5_on_the_fourth_display ();
                clock_time_change_status =
digitalRead(first_tact_button);
                if(clock_time_change_status == LOW)
                {
                  break;
                }
              }
              break;
            }
          }
          break;
        }
      }
```

```
        timing_structure.hour = hour_standing_on_the_clock;
        timing_structure.minute = minute_standing_on_the_clock;
        real_time_clock_object.setDateTime(&timing_structure);
      }
}
```