МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
канд. техн. наук, доцент
_____ Я. М. Крайник
« __ » _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Годинник Фішера для гри у шахи на базі Arduino

Спеціальність 123 Комп'ютерна інженерія
123 – КР.1 – 405.21810503

Студент _____ Булавський Д. В.
підпис

«__» _____ 2022__ р.

Керівник канд. фіз.-мат. наук, доцент _____ Дворник О. В.
підпис

«__» _____ 2022__ р.

Миколаїв – 2022

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

Факультет: комп'ютерних наук
Кафедра: комп'ютерної інженерії
Рівень вищої освіти: Перший (бакалаврський) рівень
Галузь знань: 12 Інформаційні технології
Спеціальність: 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри,
Канд. техн. наук, доцент
_____ Я.М. Крайник
« __ » _____ 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Прізвище, ім'я, по батькові: Булавський Дмитро Віталійович

1. Тема роботи: Годинник Фішера для гри у шахи на базі Arduino.

Керівник: Дворник Ольга Василівна канд. фіз.-мат. наук, доцент.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора ЧНУ ім. П.Могили від « ___ » _____ 2022__ р. № ___

2. Строк подання кваліфікаційної роботи: « 14 » червня 2022 р.

3. Вихідні дані до роботи: Розробка проекту була на мові програмування Arduino (C/C++). Із заліза були використані такі деталі: плата Arduino Uno R3, Trema-модуль чотирирозрядний LED індикатор та Trema-модуль годинник реального часу, RTC, Trema Модуль Зумер. Для них використані бібліотеки iarduino_4LED та iarduino_RTC

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Титульний аркуш, завдання на КРБ, анотація українською та англійською мовами, зміст, перелік умовних позначень, символів, одиниць та термінів (за необхідності).
Вступ. Огляд існуючих аналогів годинника. Апаратне забезпечення, яке необхідне для збору проекту.
Вибір мови програмування. Розробка алгоритму.
Програмне забезпечення годинника. Розробка коду.
Спеціальна частина з охорони праці. Висновки. Перелік джерел посилання.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень):
Функціональна схема мови Arduino.

Принципова схема підключення компонентів до Arduino.

Блок-схема алгоритму роботи годинника.

6. Консультанти розділів бакалаврської роботи:

Розділ:	Посада, науковий ступінь, вчене звання, ПІБ консультанта	Дата, підпис консультанта	
		Завдання видав	Завдання прийняв
Спеціальна частина з охорони праці	ст. викладач А. О. Алексєєва	__.__.20__. _____	__.__.20__. _____
		__.__.20__. _____	__.__.20__. _____

7. Дата видачі завдання до кваліфікаційної роботи бакалавра : « __ » _____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання	Примітки:
1.	Розробка та затвердження завдання на КРБ	21.03.2022	Виконано
2	Огляд літератури за темою роботи	23.03.2022	Виконано
3	Складання календарного плану КР	29.03.2022	Виконано
4	Аналіз предметної області	05.04.2022	Виконано
5	Розробка проектних рішень	10.04.2022	Виконано
6	Моделювання та конструювання АПЗ	20.05.2022	Виконано
7	Перевірка працездатності, тестування розробленого АПЗ, аналіз результатів тестування	01.06.2022	Виконано
8	Відгук керівника КР	03.06.2022	Виконано
9	Рецензування	05.06.2022	Виконано
10	І попередній захист	10.06.2022	Виконано
11	Оформлення кваліфікаційної роботи та презентації	15.06.2022	Виконано
12	ІІ попередній захист	20.06.2022	Виконано
13	Захист кваліфікаційної роботи	28.06.2022	Виконано

Студент _____ Булавський Д. В.
(підпис) (ініціали, прізвище)

Керівник кваліфікаційної роботи _____ Дворник О. В.
(підпис) (ініціали, прізвище)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environmen
RTC	–	(Real Time Clock) - годинник реального часу
USB	–	(Universal Serial Bus) - універсальна послідовна шина
LED	–	Light Emitting Diode
LCD	–	Liquid crystal display (Рідкокристалічний дисплей)
ПП	–	програмний продукт
Гц	–	Герц
КРБ	–	Кваліфікаційна робота бакалавра
БР	–	Бакалаврська робота
АПЗ	–	Апаратно-програмне забезпечення

АНОТАЦІЯ

до кваліфікаційної роботи
«Годинник Фішера для гри у шахи на базі Arduino»
Студент 405 гр.: Булавський Дмитро Віталійович
Керівник: канд. фіз.-мат. наук, доцент. Дворник О. В.

Бакалаврська робота присвячена розробці апаратно-програмного комплексу у вигляді годинника для гри у шахи. Актуальність теми бакалаврської роботи зумовлена появою великої кількості вбудованих пристроїв, які мають різну функціональність. Розглянуто наявні в наш час професійні та аматорські годинники. Практичне значення результатів розроблення апаратно-програмного комплексу полягає – підвищення якості навчання людей у ситуаціях коли треба швидко приймати рішення, на прикладі такої гри як шахмати.

Пояснювальна записка бакалаврської роботи складається зі вступу, двух розділів, висновків та додатків. У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет та завдання дослідження та розроблення бакалаврської роботи.

У першому розділі розглянуто апаратні компоненти, які знадобляться для розробки проекту. Проводиться аналіз ринку і технічних рішень розробки годинника. Також розглянуто характеристики компонентів та для чого вони потрібні. У другому розділі наведені дані про програмну частину комплексу та результати тестування. Четвертий розділ присвячений охороні праці щодо норм електробезпеки комп’ютерного обладнання. У висновках наведено аналіз виконаної роботи та отриманих результатів дослідження та розроблення.

В цілому, бакалаврська робота без додатків містить 56 сторінок, 24 рисунків, 2 таблиць, 20 джерел посилання.

Ключові слова: годинник Фішера, шахи, модуль годинник реального часу, летючі речовини, модуль чотирирозрядний LED індикатор, Arduino.

ABSTRACT
of the Bachelor’s Thesis
" Fisher's watch for chess based on Arduino "
Student: Bulavskyi Dmytro Vitaliiovych
Consultant: Cand. Sc., Docent. Dvornyk O. V.

The bachelor's thesis is devoted to the development of a hardware-software complex in the form of a clock for playing chess. The relevance of the topic of the bachelor's thesis is due to the emergence of a large number of built-in devices that do not have different functionality. The current professional and amateur watches are considered. The practical significance of the results of the developed hardware and software complex is - improving the quality of teaching people in situations where you need to make quick decisions, as in the example of such a game of chess.

The explanatory note of the bachelor's thesis consists of an introduction, two sections, conclusions and appendices. In the introductory question the relevance of the topic, formed the purpose, scope, subject and objectives of research and development of bachelor's work.

The first section discusses the hardware components that will be needed to develop the project. The analysis of the market and technical decisions of development of watch is carried out. The characteristics of the components and why they are needed are also considered. The second section provides data on the software part of the complex and test results. The fourth section is devoted to labor protection in relation to electrical safety standards of computer equipment. The research provides an analysis of the work performed and the results of research and development.

In total, the bachelor's thesis without appendices contains 56 pages, 24 figures, 2 tables, 20 reference sources .

Keywords: Fisher's clock, chess, real-time clock module, summer substances, four-digit LED indicator module, Arduino.

ВСТУП

Сотні років тому з’явився сучасний вид шахів до якого ми вже звикли і бачили тисячу разів. Для когось це бессмертна классика, а для когось це нудна гра в якій немає нічого, що би захоплювало дух. У середньому, аматори та професійні гравці грають одну партію приблизно дві або три години. Є люди котрі не можуть дивитися на інтелектуальну баталію стільки часу та вони почали вигадувати нові різновиди шахів. Один із таких видів – це швидкі шахи. Шахи, в які люди роблять швидкі ходи від 5 секунд до 1 хвилини.

Актуальність такого годиннику зумовлюється популярністю серіалу “The Queen's Gambit”, який вийшов у 2020 році. Після цього серіалу мільйони людей зацікавились шахами там почали в них розбиратися. Також в цьому серіалі були показані «швидкі шахи», де герої грали одну партію за 30 секнуд і на кожен крок гравців уходило не більше 5 секунд і партія могла закінчитися за 30-60 секунд. Це підігрівало азарт до цієї гри і показало, що шахмати можуть бути видовищними. Також можна зазначити, що такий вид шахів сприяє розвитку мислення та швидкого прийняття рішення, що буде корисним для кожної людини. Але для таких шахів потребувався особливий годинник з яким буде легко та зручно контролювати час.

Мета: розробити апаратно-програмний комплекс годинника для контролю часу та багатофункціональністю на базі Arduino.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

– з аналітичного огляду літератури та патентної інформації порівняти переваги та недоліки існуючих системи;

– обґрунтувати вибір складових, комплектуючих та технологій для розроблення;

– розробити алгоритм роботи пристрою для вимірювання часу;

– розробити програмне забезпечення приладу для моніторингу часу на практиці;

– оцінити перспективи подальших розроблень та удосконалень;

Об’єкт: Технології обміну повідомленнями між вбудованими модулями.

Предмет: Система відображення у реальному часі годинника та модулів.

Практичне значення: Підвищення якості навчання людей у ситуаціях коли треба швидко приймати рішення, на прикладі такої гри як шахмати.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ ТА ІНФОРМАЦІЇ ЩО ДО КОМПОНЕНТІВ ПРОЕКТУ

Моїм завдання є створення годинника Фішера для швидких шахів на базі Arduino. Спочатку я розібрався для чого потрібен годинник Фішера, як він працює, та які функції він має.

Годинник Фішера – різновид шахового годинника, запропонований та запатентований одинадцятим чемпіоном світу з шахів Робертом Фішером.

«Годинник Фішера» заснований на ідеї зрівняти шанси суперників наприкінці гри, коли в цейтноті «висить прапорець». Тим, хто грав партії з шаховими годинниками, знайоме почуття досади, коли через брак часу виграшна позиція обертається нічиєї або навіть поразкою. «Годинник Фішера» додає певну кількість секунд за кожний зроблений хід. Таким чином, якщо укладатися в цей ліміт, то «прапорець» на годиннику ніколи не впаде. Більш того, невитрачені секунди накопичуються, і замість зменшення часу, що залишився може відбутися його прибавка.

Після аналізу джерел я виявив особливості годинника та як він має працювати:

- 5 різних алгоритмів, що охоплюють усі популярні стандарти часу
- Режими збільшення часу попереднього переміщення
- Режими збільшення часу після переміщення
- До 3 різних визначень стадії за гру.
- Режим часу без збільшення
- Сукупний режим додавання
- Несукупний режим додавання
- Переміщує контроль для кожного етапу.
- Переміщення лічильника для всіх параметрів

- Призупинити/Відновити гру

1.1 Аналіз ринку годинників

Проаналізувавши СНД ринок я не знайшов жодного аналогу моєму проекту. Є тільки стандартні часи з відліком часу без різних режимів, які є у мене. Годинники такого типу потрібні для професійних та аматорських турнірів. Ціна таких годинників знаходяться у межі від 500 до 1500грн.

Рисунок 1.1.1 – Механічний годинник

Різняться вони у розмірі самого годинника та матеріалом із якого вони зроблені. Далі я звернув увагу на західний ринок. Ситуація там так ж як і в СНД. Пошук я проводив на сайті ebay. Жодного годинника я не знайшов схожий по функціоналу на мій. Для прикладу я взяв годинник шаховий електронний FLOTT F908 - gsport

Рисунок 1.1.2 – Годинник шаховий електронний FLOTT F908

Годинник є електронним табло, поміщене в корпус з міцного пластику. Воно розділене на дві частини, які містять інформацію про

поточний час і ліміт для кожного гравця. У годиннику є кілька встановлених режимів - по кілька для різних ігор, у тому числі для шахового. При цьому користувач може змінити налаштування на свій розсуд, для цього над електронним табло розміщена панель керування. Зверху годинника є клавіша, за допомогою якої гравець, зробивши хід, зупиняє свій годинник і запускає годинник противника.

Переваги :

- Висока якість виготовлення забезпечує їх точність та довговічність.
- Підходять як для аматорських ігор, так і професійних турнірів.
- Живлення від звичайних батарейок, які є в будь-якому супермаркеті.

Недоліки:

1) Має один режим роботи.
2) Треба мати з собою запасну батарейку.

У підсумку можна сказати, що західні годинники відрізняються від СНД тим, що на заході більше електронних годинників, а в СНД годинники по більшості це механічні.

1.2 Апаратні компоненти

В якості програмованої платформи вирішено обрати один з мікроконтролерів родини Arduino. Arduino підходить, як початківцям так і професіоналам та являє собою електронний конструктор з зручною платформою швидкої розробки електронних пристроїв.

Саме Arduino обрано через : доступність, зручність, відкриту архітектуру та програмний код, простоту мови програмування. Arduino дає змогу комп'ютеру взаємодіяти з фізичним світом, виходячи за рамки віртуального. Пристрої на базі Arduino можуть управляти іншими виконавчими пристроями, або використовуючи різні датчики, отримувати інформацію про навколишнє середовище. Програмування пристрою відбувається через USB без використання програматорів.

Незважаючи на те, що на ринку існує безліч різновидів Arduino, найчастіше використовуються як інженерами, так і любителями:

- Arduino UNO
- Arduino Nano

Різниця між Arduino UNO та Arduino Nano:

Плата Arduino Nano схожа на плату Arduino UNO, включаючи подібний мікроконтролер, такий як Atmega328p. Таким чином, вони можуть використовувати подібні програми. Основна різниця між ними - це розмір.

Оскільки розмір Arduino Uno вдвічі більший за наноплату. Тож дошки Uno використовують більше місця в системі. Програмування UNO можна виконати за допомогою кабелю USB, тоді як Nano використовує кабель mini USB. Основні відмінності між Uno та Nano перелічені у таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця Arduino UNO та Arduino Nano

Specification	**Arduino Uno**	**Arduino Nano**
Processor	ATmega 328P	ATmega 328P
Input Voltage	5V/7-12V	5V/7-12V
Speed of CPU	16 MHz	16MHz
Analog I/O	6/0	8/0
Digital IO/PWM	14/6	14/6
EEPROM/SRAM [kB]	1/2	1/2
Flash	32	32
USB	Regular	Mini
USART	1	1

Спираючись на розміри, для годинника Фішера, обрано Arduino Uno.

1) Arduino Uno R3 – 1шт. Характеристики:
   - Мікроконтролер: ATmega328
   - Розрядність: 8 біт
   - Напруга живлення: 5 В

– Вхідна напруга (рекомендована): 7-12 В

– Вхідна напруга (гранична): 6-20 В

– Цифрові висновки I/O: 14 ліній (6 з них – ШІМ)

– Аналогові входи: 6 (АЦП)

– Максимальний струм на виведенні I/0: 20 мА (для кожного виводу)

– Максимальний струм на виведенні 3,3V: 50 мА

– Flash-пам'ять: 32 Кб (з них 0.5 Кб використовуються під завантажувач)

– SRAM-пам'ять: 2 Кб

– EEPROM-пам'ять: 1 Кб

– Тактова частота: 16 МГц

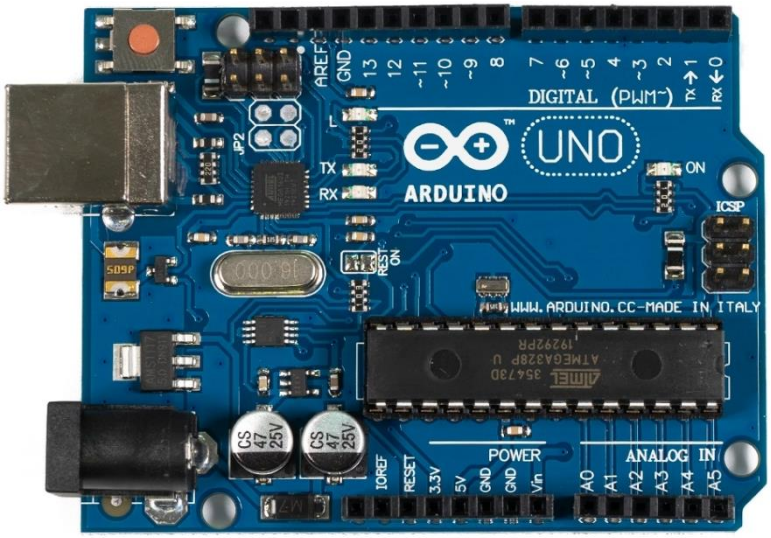

Рисунок 1.2.1 – Arduino Uno R3

2) Trema Set Shield – 1шт.

Trema Set Shield – це плата розширення, яка спрощує процес підключення модулів до Arduino . Використовуючи цю плату, ви можете створювати пристрої без проводів. Посилені роз'єми, призначені для встановлення Trema модулів, мають збільшений термін служби і забезпечують надійне електричне з'єднання контактів. Trema Set Shield має 6

секцій для встановлення модулів Trema v2.0. Верхня колодка кожної секції призначена для підключення модулів до шини I2C (всі 6 модулів, встановлених на платі, можуть бути підключені до однієї шини I2C). Нижня колодка кожної секції призначена для підключення модулів до різних цифрових чи аналогових висновків Arduino (номери висновків Arduino вказані поруч із колодкою). У центрі кожної секції написано які саме Trema-модулі можна встановлювати на дану секцію. Наявність всього двох колодок у секції не дозволить Вам неправильно встановити модуль, т.к. при неправильному підключенні його буде зміщено щодо розмітки своєї секції.

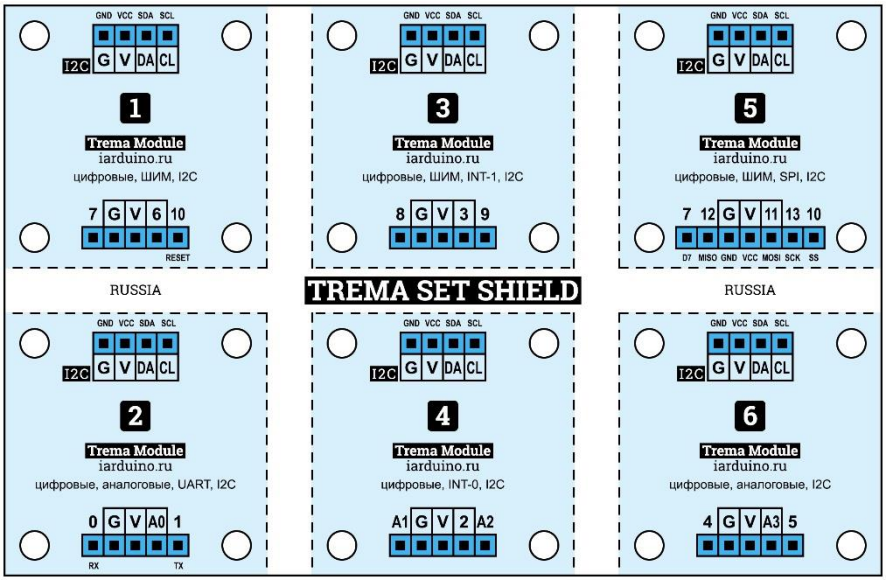

Рисунок 1.2.2 – Trema Set Shield

3) Trema-модуль Зуммер – 1шт.

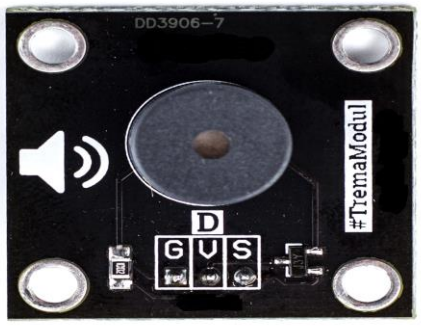

Рисунок 1.2.3 – Trema-модуль Зумер

Активний зумер

Модуль звуку, звуковипромінювач для Arduino використовується у нескладних проектах на мікроконтролерах для забезпечення звукової сигналізації будь-якої функції, процесу тощо.

"Активний" означає, що в буззері є свій вбудований звуковий генератор і він здатний працювати при подачі на нього напруги живлення.

Зумер керується Arduino контролером або іншим керуючим мікропроцесорним пристроєм за допомогою спеціальних програм та бібліотеки TONE.

Характеристики

- Напруга живлення: 5 В
- Споживаний струм: до 30 мА
- Інтенсивність звуку: >= 85 дБ
- Резонансна частота: 2048 Гц
- Опір обмотки: 40 Ом
- Робоча температура: -20 ... 70 °C
- Габарити: 30x30x9

4) Trema-модуль годинник реального часу, RTC – 1шт.

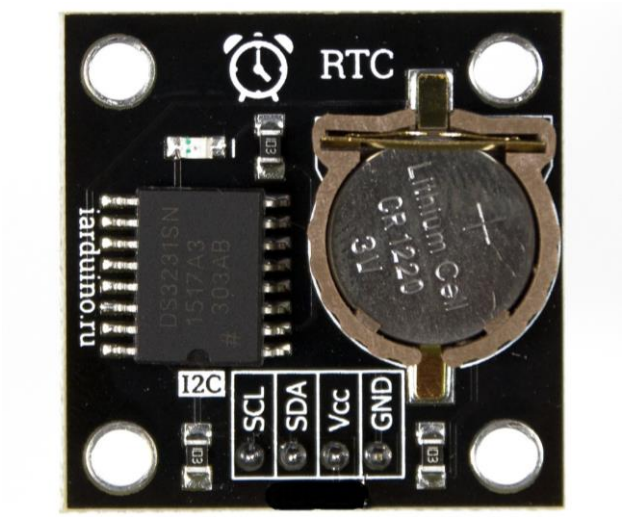

Рисунок 1.2.4 – Trema-модуль годинник реального часу, RTC

Модуль годинника реального часу - це електронна схема, призначена для обліку хронометричних даних (поточний час, дата, день тижня та ін), являє собою систему з автономного джерела живлення та пристрою, що враховує.

Модуль DS3231 по суті є звичайним годинником. У платах Arduino вже є вбудований датчик часу *Millis* , проте він працює лише при поданому живленні на плату. При відключенні та подальшому включенні Arduino відлік Millis скинеться до нуля. А DS3231 має на борту батарейку, яка навіть за відключеної плати Arduino продовжує «живити» модуль, дозволяючи йому вимірювати час.

Модуль можна використовувати як годинник або будильник, побудований на базі плат Arduino. Або ж як оповіщення для різних систем, наприклад в «Розумному домі».

Характеристики

– Чіп DS3231

– Живлення модуля: 3,3 або 5 (обидві напруги входять в діапазон допустимих значень).

– Споживаний струм (в режимі очікування): до 170 мкА.

– Споживаний струм (під час передачі): до 300 мкА.

– Струм (під час резервного живлення, без передачі даних): до 3,5 мкА.

– Тактова частота I2C: до 400 кГц.

– Рівень "0" на шині I2C: -0,3 ... 0,3 * Vcc Ст.

– Рівень "1" на шині I2C: 0,7 * Vcc ... Vcc + 0,3 Ст.

– Напруга живлення батареї: 2,3..5,5 В (номінально 3,0 В).

– Робоча температура: 0...70 °C.

– Точність ходу: ±2 ppm (приблизно ± 1 хвилина на рік)

– Габарити 30x30 мм.

5) Trema-модуль кнопка зі світлодіодом, синя – 1шт.

Trema-модуль Кнопка - це тактова кнопка, яка може бути джерелом сигналів (команд) для Ваших проектів. Кнопки використовуються для керування пристроями, подачі команд, здійснення налаштувань, введення даних і т.д.

Trema-модуль Кнопка зі світлодіодом - це аналог Trema-модуля Кнопка , але з можливістю керування підсвічуванням самої кнопки.

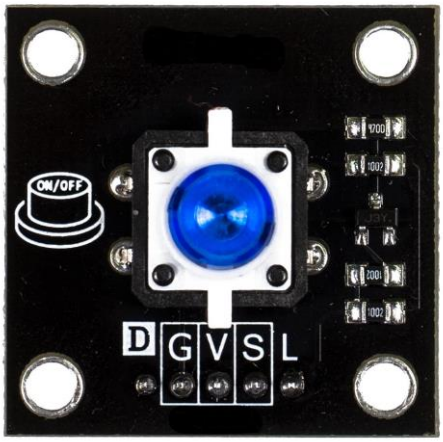

Рисунок 1.2.5 – Trema-модуль кнопка зі світлодіодом, синя

Характеристики

– Напруга живлення: від 3 до 5,5 В

– Габарити: 30×30 мм

6) Trema-модуль кнопка зі світлодіодом, червона – 1шт.

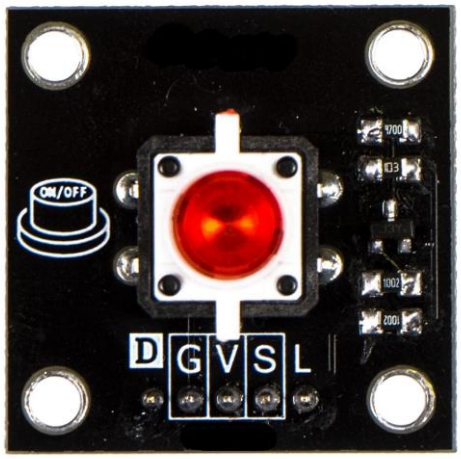

Рисунок 1.2.6 – Trema-модуль кнопка зі світлодіодом, червона

Trema-модуль Кнопка - це тактова кнопка, яка може бути джерелом сигналів (команд) для Ваших проектів. Кнопки використовуються для керування пристроями, подачі команд, здійснення налаштувань, введення даних і т.д.

Trema-модуль Кнопка зі світлодіодом - це аналог Trema-модуля Кнопка , але з можливістю керування підсвічуванням самої кнопки.

Характеристики

– Напруга живлення: від 3 до 5,5 В
– Габарити: 30×30 мм

7) Trema-модуль чотирирозрядний LED індикатор – 1шт.

Рисунок 1.2.7 – Trema-модуль чотирирозрядний LED індикатор

Модуль побудований на базі чіпа TM1637 та чотирирозрядного індикатора. Він дозволяє регулювати яскравість свічення, виводити числа (цілі, дробові, позитивні, негативні) та символи ("abcdefghijlnopstu.,:;*-_").

Для роботи з модулем, потрібно встановити бібліотеку iarduino_4LED , вона дозволяє виводити числа, час, температуру та текст.

Характеристики

– Напруга живлення Vcc: 5 ±10%
– Рівень логічної "1" на вході: > 0,7 Vcc
– Рівень логічного "0" на вході: < 0,3 В
– Споживаний струм: < 60 мА, при Vcc = 5В, всі сегменти включені на максимальну яскравість
– Споживаний струм: < 200 мкА, при Vcc = 5В, всі сегменти вимкнені
– Колір сегментів: червоний
– Робоча температура: -40...85 °C
– Габарити: 65x30x13 мм (без урахування колодки висновків)
– Вага: 20 г

Для роботи з модулем знадобиться бібліотека iarduino_4LED , що дозволяє виводити на індикатор числа, текст, масиви, час та температуру.

1.3 Схема зборання апаратного забезпечення

Встановлюємо Trema Set Shield в Arduino Uno .

Встановлюємо Trema-модуль Чотирьохрозрядний LED індикатор в другий слот поскадкового майданчика.

Рисунок 1.3.1 – Встановлення чотирьохрозрядного індикатора

Встановлюємо Trema-модуль годинник реального часу, RTC в 3 посадковий майданчик, у верхню I2C колодку.

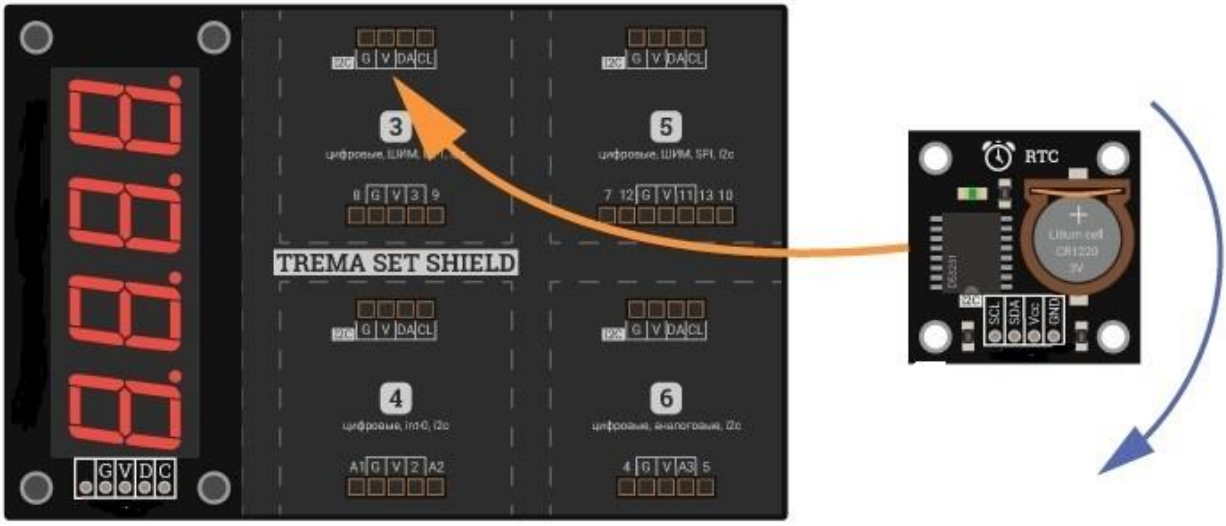

Рисунок 1.3.2 – Встановлення Trema-модуль годинник реального часу, RTC

Встановлюємо Trema-модуль Зуммер у 4 посадковий майданчик.

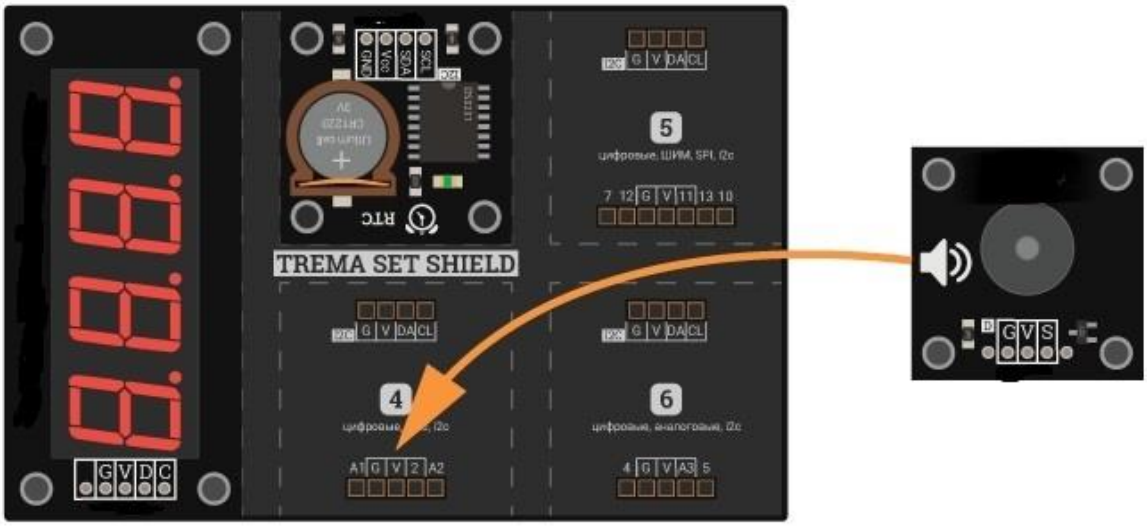

Рисунок 1.3.3 – Встановлення Trema-модуль Зуммер

Встановлюємо Trema-модуль кнопку зі світлодіодом, синя в 5 посадочні майданчики.

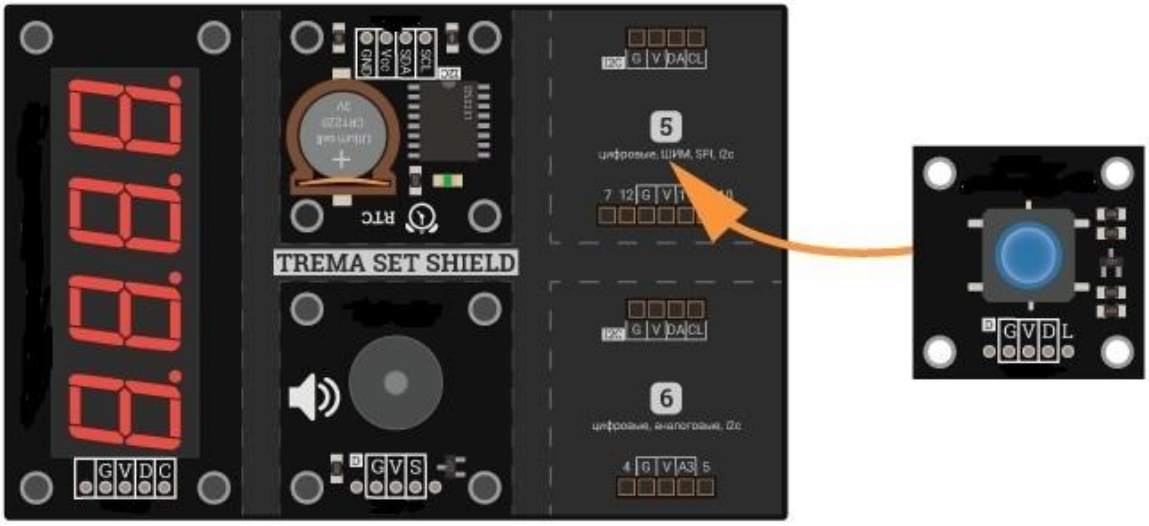

Рисунок 1.3.4 – Встановлення Trema-модуль кнопку зі світлодіодом

Встановлюємо Trema-модуль кнопку зі світлодіодом, червона в 6 посадочні майданчики.

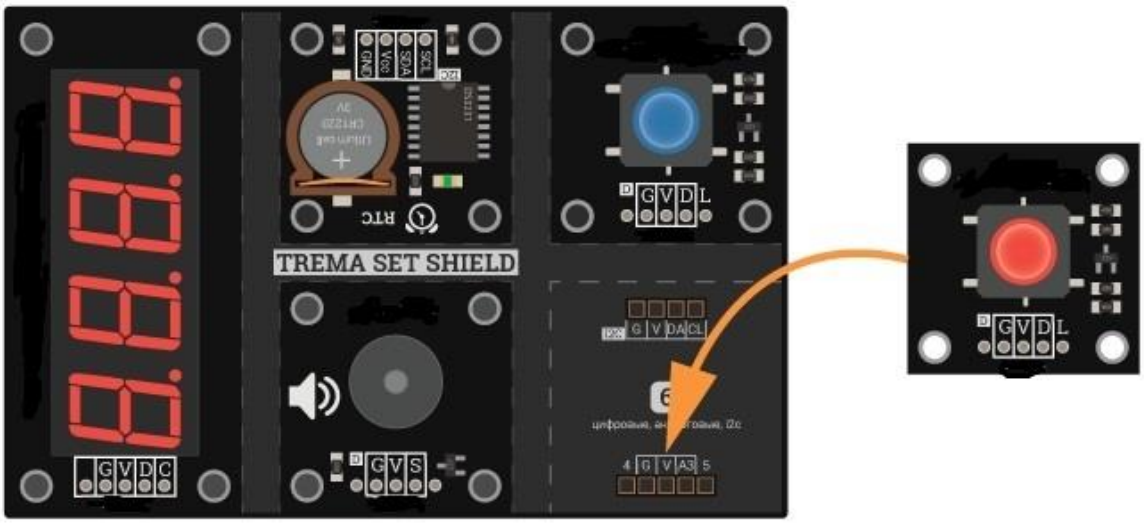

Рисунок 1.3.5 – Встановлення Trema-модуль кнопку зі світлодіодом

Отримані результати представлені нижче на рисунку.

Рисунок 1.3.6 – Результат збору

1.4 Висновки до розділу 1

У розділі 1 проаналізовано основні характеристики моделей конкурентів на ринку. Встановлено, що аналогів на ринку немає. Усі товари, що є на ринку це стандартні годинники для аматорських та професійних змагань. Вони мають тільки один функціонал – це відлік часу з контролем його для кожного гравця.

Оптимальний набір для подальшого розроблення складається з плати Arduino Uno, Trema set shield, Trema-модуль чотирирозрядний LED індикатор та Trema-модуль годинник реального часу, RTC оскільки ці показники є основними для розробки основних функцій годинника.

За допомогою порівняльного аналізу, відповідно до необхідних характеристик та прийнятній вартості, обрано комплектуючі: мікроконтролер Arduino Uno, Trema set shield, Trema-модуль чотирирозрядний LED індикатор, Trema-модуль годинник реального часу, RTC, Trema-модуль кнопку зі світлодіодом 2шт.

РОЗДІЛ 2
МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ АПЗ

2.1 Мова програмування та вибір технологій

Так як у даному проекті використовується плата Arduino Uno, для даного проекту обрано середовище розробки Arduino (IDE) (рис.2.1).

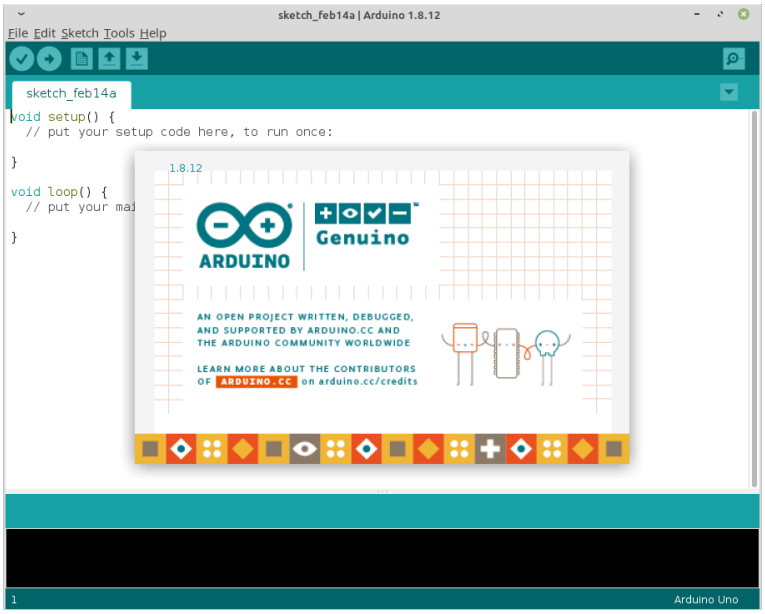

Рисунок 2.1 – Arduino (IDE)

Плата Arduino підключається до комп'ютера через USB, де вона підключається до середовища розробки Arduino (IDE).

Через простоту програм, які написані за допомогою IDE Arduino IDE, їх називаються скетчами. По суті, це текстові файли, написані мовою Arduino. Щоб зберегти та завантажити їх на плату Arduino, необхідно використовувати розширення .ino.

Є три основні частини, з яких складається мова програмування Arduino. Перш за все, функції, які дозволяють керувати дошкою. За допомогою функцій можливо аналізувати символи, виконувати математичні операції та виконувати різні інші завдання - наприклад, digitalRead () та digitalWrite () дозволяють читати або писати значення.

Кожен скетч, написаний мовою Arduino, містить дві функції. Це setUp () та loop (). Скетч завжди починається з setUp (), який виконується один раз після увімкнення або скидання плати. Це місце ідеально пілходить для ініціалізації (завдання початкових значень) змінних, установки режимів пінів (введення/виведення), завдання відповідності підключенних датчиків. Після виконання функції setup йде циклічний виклик функії loop (тобто відразу після виходу із функції setup, виконується функція loop, після виходу з неї, вона ж викликається знову. Процес триває поки живлення не буде відключено.)

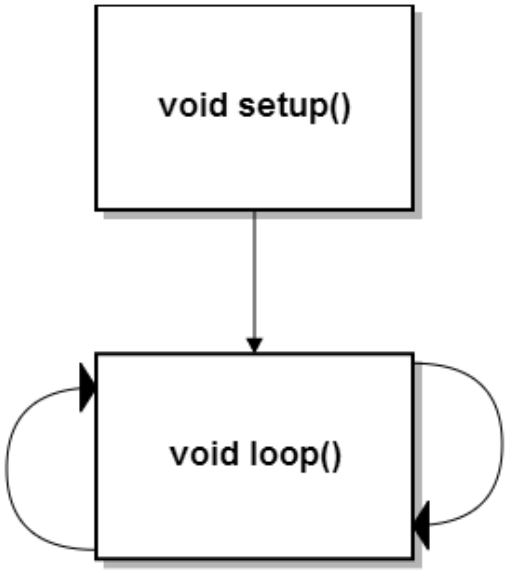

Рисунок 2.2 – Функція setup та loop

Також є значення Arduino, які представляють константи та змінні. Більшість типів даних (array, bool, char, float тощо) подібні до тих, що існують у C ++. Можливо виконати перетворення типу. Остання частина мови ардуїно називається структурою. Вона містить невеликі елементи коду, такі як оператори.

2.2 Синтаксис програмування

Що стосується синтаксису, то він не схожий на синтаксис C ++. Перша подібність, яку можливо помітити − це використання фігурних дужок для обгортання блоків коду. Якщо пропустити фігурну дужку, що закривається,

після використання тієї, що відкривається, система видасть помилку. Arduino IDE виділить закриваючу дужку, якщо натиснути на відкриваючу, так що це досить проста річ для перевірки. Як і C ++, Arduino також вимагає закінчувати твердження крапкою з комою. Пропущення призводить до спрацьовування помилки.

Ще одна чітка подібність - це спосіб введення коментарів. Існує два способи зробити це мовою Arduino, залежно від того, чи потрібен однорядковий чи блокований коментар. Якщо потрібно прокоментувати лише один рядок, необхідно розпочати його з двох похилих рисок:

```
// a comment here
#define LED_PIN
void setup() {
pinMode(LED_PIN, OUTPUT);
```

Якщо одного рядка замало для приміток, можливо вставити багаторядковий коментар, починаючи його косою рискою та зірочкою, закінчуючи зірочкою та косою рискою:

```
/* a comment here
a cemment there
there are comments everywhere */
#define LED_PIN
void setup() {
pinMode(LED_PIN, OUTPUT);
```

Додаючи коментарі, необхідно пам’ятати, що компілятор Arduino повністю їх ігноруватиме. Це означає, що він не експортуватиме їх у процесор і не використовуватиме пам’ять мікроконтролера.

2.3 Функції часу

Найпростішою з погляду використання функцією часу є затримка: програма “зависає” всередині функції затримки і чекає вказаний час. Затримка дозволяє дуже зручно та наочно організувати роботу простої “однозадачної” програми, у нас є два варіанти затримок:

– delay (time)

– Затримка на вказану кількість мілісекунд (мс). 1 секунда = 1000 мс.

– *Time* приймає тип даних unsigned long і може призупинити виконання терміном від 1 до 4 294 967 295 мс (~50 діб) з роздільною здатністю 1 мс.

– Працює на системному таймері, тому не працює всередині переривання та при відключених перериваннях.

– delayMicroseconds (time)

– Затримка на вказану кількість мікросекунд (мкс). 1 секунда = 1000000 мкс.

– Time приймає тип даних unsigned int і може призупинити виконання терміном від 4 до 16383 мкс (так, менше ніж максимум для цього типу даних) з роздільною здатністю 4 мкс.

– Працює не на таймері, а на пропуску тактів процесора, тому може працювати у перериванні та при відключених перериваннях.

– Іноді не зовсім коректно працює зі змінними, тому потрібно намагатися використовувати константи (const чи просто число).

– Часто використовують у бібліотеках для емуляції цифрових інтерфейсів зв'язку.

Затримки використовувати дуже просто:

```
void setup () {}
void loop () {
 // щось виконати
  delay ( 500 ) ; // зачекати півсекунди
}
```

Мислення “затримками” – головна проблема новачків. Організувати роботу складної програми за допомогою затримки неможливо, тому далі розглянемо корисніші інструменти.

Функція yield()

Розробники Arduino подбали про те, щоб функція delay() не просто блокувала виконання коду, а й дозволяла виконувати інший код під час цієї затримки. Данна "милиця" отримала назву yield() і працює наступним чином: якщо оголосити функцію

```
void yield () {
 // ваш код
}
```

то розташований у ній код буде виконуватися під час роботи будь-якої затримки delay () в програмі! Це рішення хоч і здається безглуздим, але в той же час дозволяє швидко і без написання зайвих милиць і таймерів реалізувати пару завдань, що паралельно виконуються. Це цілком відповідає ідеології Arduino – максимально проста та швидка розробка прототипу. Розглянемо простий приклад: стандартний миготливий світлодіод, але з опитуванням кнопки:

```
void setup () {
  pinMode ( 13, OUTPUT ) ;
}
void loop () {
  digitalWrite ( 13, 1 ) ;
  delay ( 1000 ) ;
  digitalWrite ( 13, 0 ) ;
  delay ( 1000 ) ;
}
void yield () {
 // тут можна опитувати кнопку
 // і не пропустити натискання із-за delay!
}
```

2.4 Програмна реалізація АПЗ

Закінчивши пошук необхідних деталей наступний крок – це програмна частина проекту. Після установки IDE для роботи треба встановити конкретні бібліотеки з необхідними функціями, а саме:

– Бібліотека iarduino_4LED для роботи з модулем Trema чотирьох розрядним LED індикатором .

– Бібліотека iarduino_RTC для роботи з Trema-модуль годинами реального часу, RTC .

Якщо коротко, бібліотека - це набір заздалегідь написаного коду, який надає вам додаткові функції.

Структура бібліотеки:

– бібліотека - це папка, що складається з файлів з файлами коду C ++ (.cpp) і файлів заголовків C ++ (.h);

– файл .h описує структуру бібліотеки і оголошує все її змінні і функції;

– файл .cpp містить реалізацію функції.

Багато скетчів (програм) працюють із бібліотеками. Бібліотека полегшує роботу з певним модулем або одним із типів модулів. Наприклад, якщо ви хочете вивести текст на LCD дисплей без підключення бібліотеки, то вам потрібно передати йому кілька байт команд та даних, що займе кілька рядків коду, а головне, що потрібно знати тип мікроконтролера під керуванням якого працює LCD дисплей, призначення команд якими він управляється, знає архітектуру його пам'яті, адресу і призначення регістрів, для чого потрібно знайти і перечитати його дані. Щоб включити певну бібліотеку у свій скетч, необхідно використати оператор #include і викликати бібліотеку, яку потрібно використовувати. Важливо пам’ятати, що не слід додавати крапку з комою: це твердження не потрібно припиняти.

Щоб підключити бібліотеку, потрібно написати лише один рядок на початку скетчу: "#include <файл.h>", наприклад:

#include <iarduino_4LED.h>

2.4.1 Бібліотека iarduino_4LED

Бібліотека дозволяє регулювати яскравість світла LED індикатора, виводити на нього числа (цілі, дробові, позитивні, негативні), символи ("abcdefghijlnopstu.,:;*-_"), масиви чисел, час та температуру.

Призначення функцій та змінних:

Детальний опис роботи з бібліотекою знаходиться в розділі Wiki Чотирирозрядний індикатор .

#include <iarduino_4LED.h> // Підключаємо бібліотеку.

iarduino_4LED ОБ'ЄКТ (ВИСНОВОК_CLK , ВИСНОВОК_DIO); // Створюємо об'єкт.

функція begin(); // Ініціалізація роботи з індикатором LED.

Функція clear(); // Очищення індикатора (вимикання всіх сегментів).

Функція light (ЧИСЛО); // Встановлення яскравості індикатора, від 0 до 7.

Функція point (ПОЗИЦІЯ, СТАН); // Управління точками на індикаторі.

Функція print(ЗНАЧЕННЯ [, ПАРАМЕТРИ_ВИСНОВКУ_ЧИСЛА]); // Виведення значення на індикатор.

Функція setLED([[[[БАЙТ_№1]], БАЙТ_№2], БАЙТ_№3], БАЙТ_№4], ФЛАГ]); // Встановлює світлодіоди (сегменти) індикатора з біт.

2.4.2 Бібліотека iarduino_RTC

Бібліотека дозволяє читати та записувати час RTC модулів на базі чіпів: DS1302, DS1307 , DS3231 , ...

Перевагою даної бібліотеки є зручна реалізація отримання часу.

Призначення функцій та змінних:

#include **<iarduino_RTC.h>** // Підключаємо бібліотеку. iarduino_RTC **ОБ'ЄКТ (** НАЗВА [, ВИСНОВОК_RST [, ВИСНОВОК_CLK [, ВИСНОВОК_DAT]]] **));** // Створюємо об'єкт.

функція **begin();** // Ініціалізація роботи RTC модуля.

Функція **settime(** СЕК [, МІН [, ГОДИННИК [, ДЕНЬ [, МІС [, РІК [, ДН]]]]]] **);** // Встановлення часу.

Функція **gettime(** [РЯДОК_ШАБЛОН] **);** // Читання часу.

Функція **blinktime (** ПАРАМЕТР [ЧАСТОТА] **);** // Примушує функцію gettime "блимати" вказаним параметром часу.

Функція **period(** ХВИЛИНИ **);** // Вказує мінімальний період звернення до модуля хвилинах.

Функція **settimeUnix(** СЕКУНДИ **);** // Встановлення часу Unix time.

функція **gettimeUnix();** // Читання часу Unix time.

При будь-якому зверненні до функцій gettime() та gettimeUnix(), автоматично оновлюються значення наступних змінних:

Змінна **seconds** // Містить секунди від 0 до 59.

Змінна **minutes** // Містить хвилини від 0 до 59.

Змінна **hours** // Містить годинник від 1 до 12.

Змінна **Hours** // Містить годинник від 0 до 23.

Змінна **midday** // Містить опівдні 0 або 1 (0-am, 1-pm).

Змінна **day** // Містить день від 1 до 31.

Змінна **weekday** // Містить день тижня від 0 до 6 (0-неділя, 6-субота).

Змінна **month** // Містить місяць від 1 до 12.

Змінна **year** // Містить рік від 0 до 99.

Змінна **Unix** // Містить секунди з початку епохи Unix, від 0 до 4'294'967'295.

2.5 Функції які були використані

Підключення бібліотек

#include <iarduino_4LED.h> // Підключаємо бібліотеку iarduino_4LED.

iarduino_4LED dispLED (1 , A0); // Оголошуємо об'єкт роботи з функціями бібліотеки iarduino_4LED, із зазначенням висновків дисплея (CLK , DIO). //

#include <iarduino_RTC.h> // Підключаємо бібліотеку.

Константа для регулювання часу.

const uint8_t fisherTime = 20 ;// Константа визначає кількість часу, яке додається після кожного натискання.

const uint8_t startGameTime = 30 ; // Константа визначальна загальна кількість часу на початку гри. //

Функція старту гри та показу таймера.

void StartGame (); // Функція початку гри.

void ShowTime (); / / Функція показу часу.

Функія додавання часу для кожного гравця.

void ConversionTimeLeft ();// Функція додавання часу фішера для лівого гравця.

void ConversionTimeRight (); // Функція додавання часу фішера для правого гравця. //

Ініціюємо LED дисплей.

dispLED . begin (); // Ініціюємо LED дисплей.

dispLED . light (7); // Встановлюємо максимальну яскравість індикатора LED.

Задаємо подію 1.

time . begin (); // Ініціюємо RTC модуль.

menu = 1 ; // Задаємо подію 1. } // // void loop () // { //

time . gettime ();// Читаємо час, оновлюючи значення всіх змінних.

switch (menu) // Вибір події.

Початок гри.

case 1 : // Подія 1.

dispLED . print (startGameTime , 0 , POS3); // Виводимо на індикатор заданий час матчу.

if (digitalRead (pinKeyBlue) && digitalRead (! pinKeyRed)) // Перевіряємо натискання синьої кнопки та забороняємо натискання червоної кнопки.

{ //

StartGame (); // Функція початку гри.

Зменшення часу для гравця лівого гравця.

timeLeft = 59 - time . seconds ; // Зменшуємо час у лівого гравця на секунду. if (timeLeft == 58) { eventLeft = true ;}// Дозволяємо зменшення хвилин у лівого гравця на 58 секунді.

if (timeLeft == 59 && eventLeft) // Зменшуємо хвилини у лівого гравця на 59 секунді, якщо дозволено зменшення хвилин.

Фунція ініціалізації компонентів.

Void setup()

Фунція події 4.

Case 4 : // Подія 4

ShowTime (); / / Функція показу часу.

if (digitalRead (pinKeyBlue) | | digitalRead (pinKeyRed)) // Перевірка натискання синьої чи червоної кнопки.

{ menu = 1 ;} // Повертаємось у подію 1.

Фунція події 5.

Case 5 : // Подія 5.

ShowTime (); / / Функція показу часу.

if (digitalRead (pinKeyBlue) | | digitalRead (pinKeyRed))// Перевірка натискання синій чи червоної кнопки.

{ menu = 1 ;} // Повертаємось у подію 1.

digitalWrite (pinBuzzer , HIGH); // Включаємо зумер.

digitalWrite (pinKeyLedBlue , HIGH); // Вмикаємо світлодіод на блакитній кнопці.

delay (100); // Затримка 100 мс.

digitalWrite (pinBuzzer , LOW); // Вимикаємо зумер.

2.6 Опис інтерфейсів АПЗ

2.6.1 Підключення ds3231 до Arduino.

Годинник підключається по двопровідній шині IIC (Inter-IntegratedCircuit) через висновки SDA та SCL до SDA та SCL Arduino відповідно. Також необхідно підключити живлення +5В та GND. Інші висновки специфічні для DS3231 та не підтримуються бібліотеками для DS1307, та й не часто використовуються.

Таблиця 1.1 – Підключення контактів

	Контакт SDA	**Контакт SCL**
Arduino UNO, Nano, Pro Mini etc.	A4	A5
На Arduino Mega, Due	20	21
На Arduino Leonardo, Pro Micro	2	3

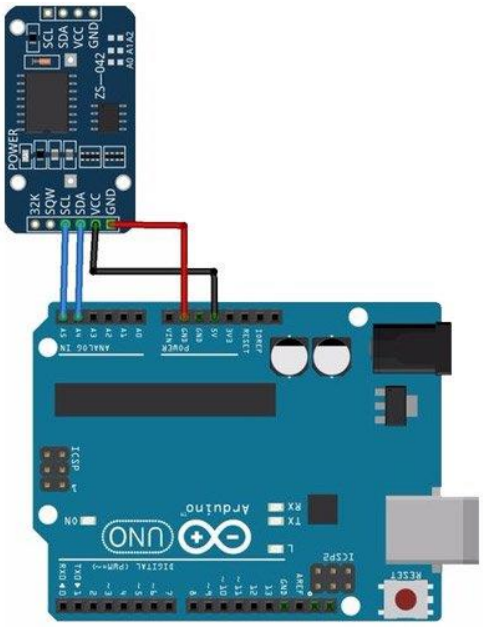

Рисунок 2.3 – Підключення ds3231 до arduino

Групи контактів - J1

32K: вихід генератора, частота 32 кГц

SQW: Вихід прямокутного (Square-Wave) сигналу.

SCL: Serial CLock – шина тактових імпульсів інтерфейсу I2C

SDA: Serial Data – шина даних інтерфейсу I2C;

VCC: "+" живлення модуля

GND: "-" живлення модуля

Групи контактів - J2

SCL: лінія тактування (Serial CLock)

SDA: лінія даних (Serial Data)

VCC: "+" живлення модуля

GND: "-" живлення модуля

2.6.2 Підключення модуля DS3231 RTC Arduino до шини I2C

Для Arduino UNO:

SCL → A5

SDA → A4

VCC → +5

GND → земля

Підключення відбувається по двопровідній шині I2C(TWI)

Висновки SDA та SCL підключаються до аналогічних висновків на Arduino

Живлення VCC до +5 Вольт, а GND до GND на платі Arduino

Для роботи необхідно встановити бібліотеку DS3231.

2.6.3 Підключення LED індикатора.

На платі модуля розташований роз'єм із 4 висновків для підключення до шини I2C.

- **SCL** – вхід/вихід лінії тактування шини I2C.
- **SDA** – вхід/вихід лінії даних шини I2C.
- **Vcc** - вхід живлення 3,3 або 5 ст.
- **GND** – загальний вихід живлення.

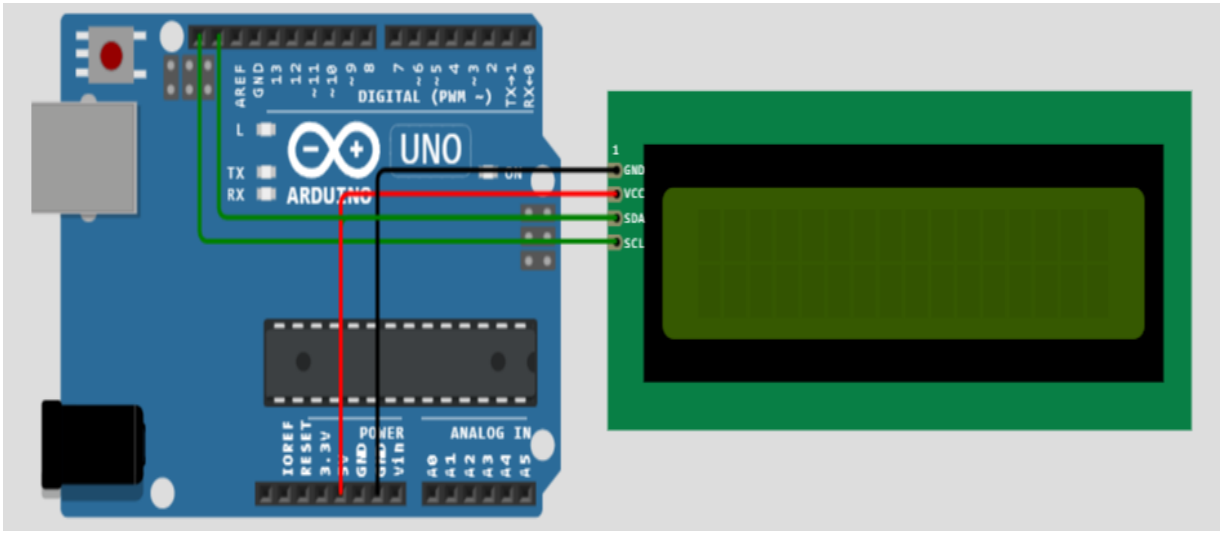

Рисунок 2.4 – Підключення LED індикатора

2.6.4 Підключення зумера до Arduino

Зумер зазвичай використовується для подачі сигналу як тонального або звукового сигналу. Цей тип зумера широко використовується в системах сигналізації, побутових приладів або у вбудованих системах, щоб забезпечити будь-яку індикацію або оповіщення.

Складається він з п'єзоелектричного матеріалу, який приклеєний до тонкої металевої пластини. Якщо подати напругу на ці пластини, п'єзоелемент почне згинатися і розгинатися, створюючи певний звук. Чим швидше вигинає п'єзоелемент, тим вище рівень шуму, ця швидкість називається частотою. Знову ж таки, чим вища частота, тим вищий звук, який ми чуємо.

Для підключення його Arduino необхідно використовувати транзистор, щоб не спалити вихід контролера. Також можна скористатися модулем, на якому вже встановлений зумер, транзистор, резистор і трьох контактний роз'єм, кроком 2.54 мм. Принципова схема показана малюнку нижче.

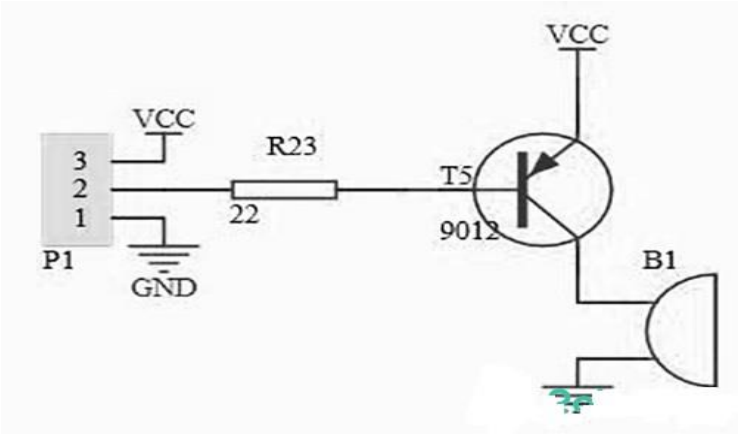

Рисунок 2.5 – Схема роботи зумера

Призначення контактів:

– VCC – напруга живлення;
– I/O – сигнал керування;
– GND – загальний контакт.

Схема не складна, необхідно всього три дроти, спочатку підключаємо шину I/0 до порту 8 (Arduino UNO), залишилося підключити живлення GND до GND і VCC до 5V (можна записати і від 3.3В)

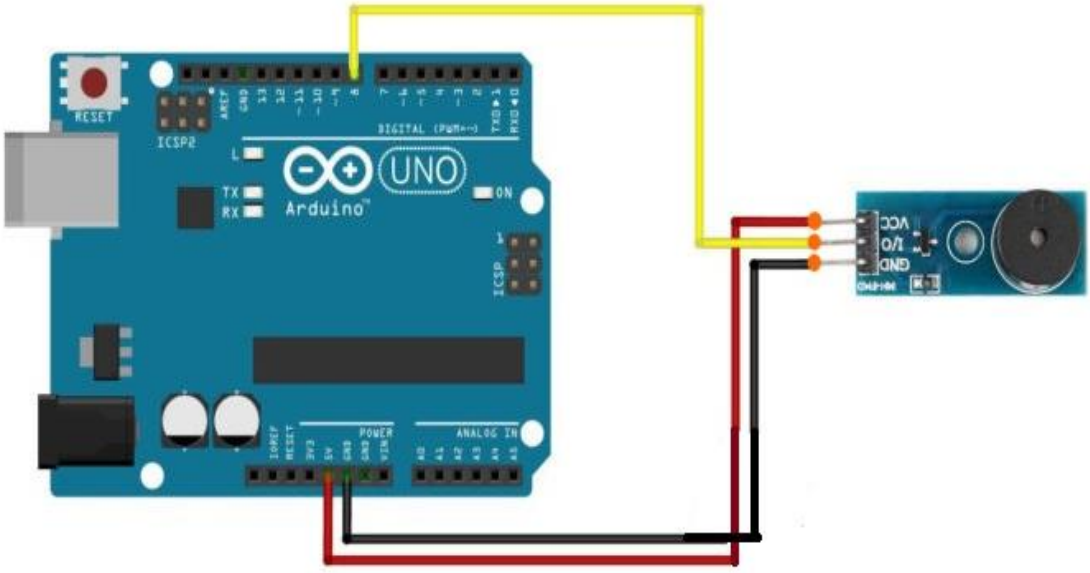

Рисунок 2.6 – Підключення зумера

Загальна схема підключення приладу для контролю часу.

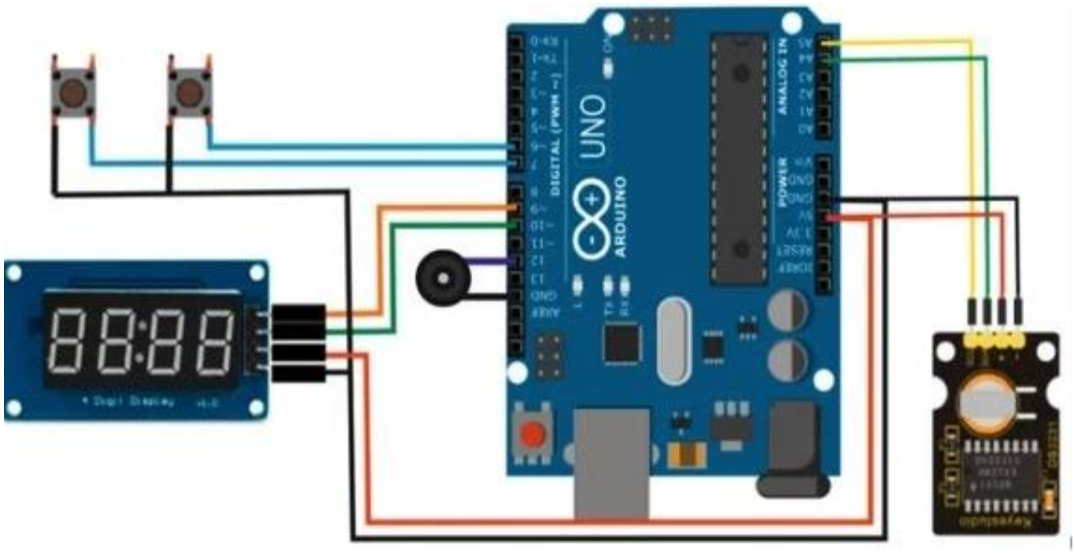

Рисунок 2.6 – Підключення усіх компонентів

2.6.5 Результат збирання приладу

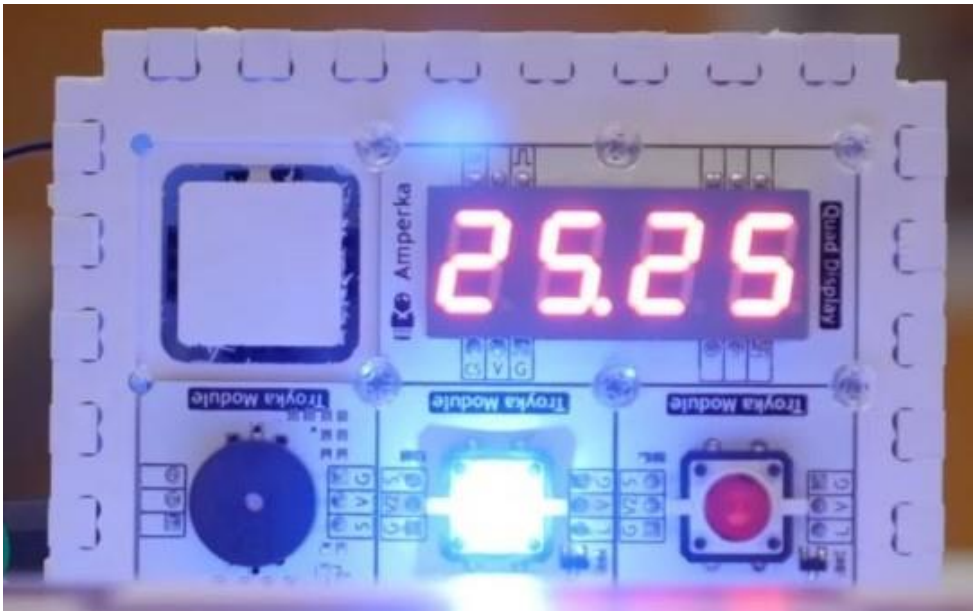

Рисунок 2.7 – Вигляд годинника Фішера

2.7 Алгоритм роботи проекту

У коді loop будуть розглянуті усі події від 1 до 5.

Читаємо час, з модуля годинника реального часу.

Подія 1.

Виводимо на екран кількість заданих хвилин.

Очікуємо натискання однієї із кнопок. При натисканні переходимо до функції "StartGame()" у якій обнулюємо початковий час відліку, зберігаємо

заданий час матчу для лівого гра та правого гравця. При натисканні червоної кнопки дозволяємо подію 2. При натисканні блакитної кнопки дозволяємо подію 3.

Подія 2 та 3. Поодинокі події для різних кнопок.

Вимикаємо зумер. Гасимо світлодіод на кнопці суперника. Вмикаємо світлодіод на кнопці.

Віднімаємо реальний час із хвилини та привласнюємо змінній часу.

Якщо 58 секунд дозволяємо зменшувати хвилини. Якщо 59 секунд зменшуємо хвилини на один, забороняємо зменшувати хвилини.

Якщо кількість хвилин менше або дорівнює нулю, дозволяємо виведення на екран секунд, що залишилися.

Якщо дозволено виведення кількості секунд, що залишилися, зберігаємо секунди в змінну хвилин, для подальшого їх виведення.

Якщо хвилини і секунди рівні нулю, то переходимо до події 4 або 5 залежно від кнопки. Для червоної кнопки перехід до події 4, синій перехід до події 5.

Перевірка натискання своєї кнопки для завершення свого ходу, але не раніше 1 секунди після ходу. Дозволяємо зменшення хвилин. Включаємо зумер. Якщо дозволено виведення кількості секунд, що залишилися, зберігаємо секунди в змінну хвилин, для подальшої з ними роботою і висновком. Зберігаємо реальний час у змінну часу. Встановлюємо реальний час для суперника. На початку гри воно дорівнює нулю. Після ходу воно дорівнює кількості реального часу, збереженого за попередній хід. Додаємо час Фішера функцією "ConversionTimeLeft()" для гравця із червоною кнопкою або "ConversionTimeRight ()" для гравця із синьою кнопкою.

Ці функції ідентичні за принципом дії, але різних кнопок. У них визначаємо залишок часу гравцю, додаємо час Фішера гравцю, перевірка отриманий час більше хвилини, якщо більше, то відсікаємо частину хвилини і зберігаємо час, що залишився: хвилини зберігаємо в змінну хвилин, секунди зберігаємо в змінну секунд. Якщо ж дозволено виведення секунд на дисплей,

значить хвилини дорівнюють нулю, тому вводимо перевірку хвилин і так само перетворимо час на хвилини та секунди, якщо кількість секунд перевищує хвилину. Визначаємо реальний час і виходимо з функцій "ConversionTimeLeft ()" або "ConversionTimeRight ()" .

Далі дозволяємо переходимо до події залежно від натиснутої кнопки. Якщо натиснута червона кнопка, переходимо до події 2, якщо натиснута синя кнопка, то переходимо до події 3, тобто настає хід другого гравця і переходимо до протилежної події.

Виводимо час на індикатор за допомогою функції "ShowTime()". У ній перетворимо час цілого значення в рядкове, додаючи нуль, де це необхідно. А також виводимо раз на пів секунди загальний час і миготливу точку по центру.

4. Подія, коли гравець за червоною кнопкою програв. Тут вмикається і вимикається зумер і світлодіод червоної кнопки, сповіщає про програш гравця, а також очікування натискання будь-якої з кнопок, щоб повернутися в подію 1, тобто розпочати гру заново.

5. Подія, коли гравець за синьою кнопкою програв. Тут вмикається і вимикається зумер і світлодіод синьої кнопки, сповіщає про програш гравця, а також очікування натиснення будь-якої з кнопок, щоб повернутися в подію 1, тобто почати заново гру.

2.8 Результат програмування проекту

```
#include <iarduino_4LED.h> // Підключаємо бібліотеку iarduino_4LED.
iarduino_4LED dispLED ( 1 , A0 ); // Оголошуємо об'єкт роботи з функціями бібліотеки iarduino_4LED, із зазначенням висновків дисплея ( CLK , DIO ). //
#include <iarduino_RTC.h> // Підключаємо бібліотеку.
iarduino_RTC time ( RTC_DS3231 ); // Оголошуємо об'єкт time для модуля з урахуванням чіпа DS3231. //
const uint8_t pinKeyRed = A3 ; // Оголошуємо пін для роботи із червоною кнопкою.
```

const uint8_t pinKeyBlue = 11; // Оголошуємо пін для роботи із зеленою кнопкою.

const uint8_t pinKeyLedRed = 5 ; // Оголошуємо пін для роботи зі світлодіодом червоної кнопки.

const uint8_t pinKeyLedBlue = 13 ; // Оголошуємо пін для роботи зі світлодіодом зеленої кнопки.

const uint8_t pinBuzzer = 2 ; // Визначаємо № виводу якого підключений зумер з вбудованим генератором. //

const uint8_t fisherTime = 20 ;// Константа визначає кількість часу, яке додається після кожного натискання.

const uint8_t startGameTime = 30 ; // Константа визначальна загальна кількість часу на початку гри. //

int menu ; // Змінна подія.

uint8_t timeLeft ; // Змінна секунд для лівого гравця.

uint8_t timeRight ; // Змінна секунди для правого гравця. uint8_t minLeft ; // Змінна хвилина для лівого гравця.

uint8_t minRight ; // Змінна хвилина для правого гравця.

String timesMin; // Змінна виведення часу в рядковому форматі. String minLeftChar ; // Змінна час для лівого гравця в рядковому форматі. String minRightChar ; // Змінна час для правого гравця в рядковому форматі. bool eventLeft = true ; // Змінна подія дозволяє/забороняє зменшення хвилин для лівого гравця.

bool eventRight = true ; // Змінна подія дозволяє/забороняє зменшення хвилин для правого гравця.

bool eventLeftZero = false ; // Змінна подія дозволяє/забороняє показ секунд для лівого гравця.

bool eventRightZero = false ; // Змінна подія дозволяє/забороняє показ секунд для правого гравця. //

bool counter = false ; // Змінна подія дозволяє/забороняє виведення на індикатор.

long prevmicros ; // Змінна для збереження попереднього значення таймера. // void StartGame (); // Функція початку гри.

void ShowTime (); / / Функція показу часу.

void ConversionTimeLeft ();// Функція додавання часу фішера для лівого гравця.

void ConversionTimeRight (); // Функція додавання часу фішера для правого гравця. //

void setup () //

{ //

pinMode (pinKeyBlue , INPUT); // Перекладаємо виведення pinKeyGreen у режим входу.

pinMode (pinKeyRed , INPUT); // Перекладаємо виведення pinKeyRed режим входу.

pinMode (pinBuzzer , OUTPUT); // Перекладаємо виведення pinBuzzer у режим виходу.

digitalWrite (

pinBuzzer , LOW); // Встановлюємо рівень логічного "0" на виведенні pinBuzzer.

dispLED . begin (); // Ініціюємо LED дисплей.

dispLED . light (7); // Встановлюємо максимальну яскравість індикатора LED.

time . begin (); // Ініціюємо RTC модуль.

menu = 1 ; // Задаємо подію 1. } // // void loop () // { //

time . gettime ();// Читаємо час, оновлюючи значення всіх змінних.

switch (menu) // Вибір події.

{ //

case 1 : // Подія 1.

dispLED . print (startGameTime , 0 , POS3); // Виводимо на індикатор заданий час матчу.

if (digitalRead (pinKeyBlue) && digitalRead (! pinKeyRed)) // Перевіряємо натискання синьої кнопки та забороняємо натискання червоної кнопки.

{ //

StartGame (); // Функція початку гри.

menu = 3 ; // Переходимо до події 3.

} //

if (digitalRead (pinKeyRed) && digitalRead (! pinKeyBlue)) // Перевіряємо натискання червоної кнопки та забороняємо натискання синьої кнопки.

{ //

StartGame (); // Функція початку гри.

menu = 2 ; // Переходимо до події 2.

} //

break ; // Виходимо з оператора

case. //

Case 2 : // Подія 2

digitalWrite (pinBuzzer , LOW); // Вимикаємо зумер.

digitalWrite (pinKeyLedBlue , LOW); // Вимикаємо світлодіод на синій кнопці, у суперника.

digitalWrite (pinKeyLedRed , HIGH); // Вмикаємо світлодіод на червону кнопку.

timeLeft = 59 - time . seconds ; // Зменшуємо час у лівого гравця на секунду. if (timeLeft == 58) { eventLeft = true ;}// Дозволяємо зменшення хвилин у лівого гравця на 58 секунді.

if (timeLeft == 59 && eventLeft) // Зменшуємо хвилини у лівого гравця на 59 секунді, якщо дозволено зменшення хвилин.

{ //

minLeft --; // Зменшуємо хвилини у лівого гравця.

eventLeft = false ; // Забороняємо зменшення хвилин у лівого гравця. } // if (minLeft <= 0) { eventLeftZero = true ;} // Якщо хвилин менше або дорівнює нулю, дозволяємо виведення на дисплей секунд у лівого гравця.

if (eventLeftZero) { minLeft = timeLeft ;} // Якщо дозволено виведення на дисплей секунд, зберігаємо значення секунд у змінну хвилин для лівого гравця.

if (timeLeft == 0 && minLeft == 0) { menu = 4 ;} // Якщо хвилини і секунди дорівнюють нулю, то переходимо до події 4. Лівий гравець програв.

if (digitalRead (pinKeyRed) && time. seconds >= 1) // Перевірка натискання червоної кнопки, але не раніше, ніж за одну секунду з початку ходу лівого гравця.

{

eventLeft = true ; // Дозволяємо зменшення хвилин у лівого гравця.

digitalWrite (pinBuzzer , HIGH); // Включаємо зумер. if (eventLeftZero) { minLeft = timeLeft ;} // Якщо дозволено виведення секунд на індикатор, зберігаємо значення секунд у змінну хвилин для лівого гравця.

timeLeft = time. seconds ; // Зберігаємо кількість секунд, що пройшли за хід у лівого гравця.

time . settime (timeRight); // Встановлюємо час у якому зупинився хід суперника. Правий гравець ConversionTimeLeft (); // Функція додавання часу фішера для лівого гравця.

menu = 3 ; // Переходимо до події 3. Хід суперника, правий гравець. } // ShowTime (); / / Функція показу часу. break ; // Виходимо з оператора case. //

Case 3 : // Подія 3

digitalWrite (pinBuzzer , LOW); // Вимикаємо зумер.

digitalWrite (pinKeyLedRed , LOW); // Вимикаємо світлодіод на червоній кнопці у суперника.

digitalWrite (pinKeyLedBlue , HIGH); // Включаємо світлодіод на синій кнопці.

timeRight = 59 - time . seconds ; // Зменшуємо час у правого гравця на секунду.

if (timeRight == 58) { eventRight =true ;} // Дозволяємо зменшення хвилин у правого гравця на 58 секунді.

```
 if ( timeRight == 59 && eventRight ) // Зменшуємо хвилини у правого гравця на 59 секунді, якщо дозволено зменшення хвилин.
 {
     minRight --; // Зменшуємо хвилини у правого гравця.
      eventRight = false ; // Забороняємо зменшення хвилин у правого гравця.
 }
 if ( minRight <= 0 ) { eventRightZero = true ;}  // Якщо хвилин менше або дорівнює нулю, дозволяємо виведення на дисплей секунд у правого гравця.
 if ( eventRightZero ) { minRight = timeRight ;} // Якщо дозволено виведення на дисплей секунд, зберігаємо значення секунд у змінну хвилин для правого гравця.
 if ( timeRight == 0 && minRight == 0 ) { menu = 5 ;} // Якщо хвилини і секунди дорівнюють нулю, то переходимо до події 4. Правий гравець програв.
 if ( digitalRead ( pinKeyBlue ) &&
     time . seconds >= 1 ) // Перевірка натискання синьої кнопки, але не раніше ніж через одну секунду з моменту початку ходу правого гравця.
 {
     eventRight = true ; // Дозволяємо зменшення хвилин у правого гравця.
     digitalWrite ( pinBuzzer , HIGH ); // Включаємо зумер.
  if ( eventRightZero ) { minRight = timeRight ;} // Якщо дозволено виведення секунд на індикатор, зберігаємо значення секунд у змінну хвилин для правого гравця.
     timeRight = time . seconds ; // Зберігаємо кількість секунд, що пройшли за перебіг у правого гравця.
     time . settime ( timeLeft ); // Встановлюємо час у якому зупинився хід суперника. Лівий гравець.
 ConversionTimeRight (); // Функція додавання часу фішера для правого гравця.
```

menu = 2 ; // Переходимо до події 2. Хід суперника, лівий гравець. } // ShowTime (); / / Функція показу часу.

break ; // Виходимо з оператора case.

Case 4 : // Подія 4

ShowTime (); / / Функція показу часу.

if (digitalRead (pinKeyBlue) | | digitalRead (pinKeyRed)) // Перевірка натискання синьої чи червоної кнопки.

{ menu = 1 ;} // Повертаємось у подію 1.

digitalWrite (pinBuzzer , HIGH); // Включаємо зумер.

digitalWrite (pinKeyLedRed , HIGH); // Вмикаємо світлодіод на червону кнопку.

delay (100); // Затримка 100 мс

digitalWrite (pinBuzzer , LOW); // Вимикаємо зумер.

digitalWrite (pinKeyLedRed , LOW); // Вимикаємо світлодіод на червоній кнопці.

delay (100); // Затримка 100 мс. break ; // Виходимо з оператора case.

Case 5 : // Подія 5.

ShowTime (); / / Функція показу часу.

if (digitalRead (pinKeyBlue) | | digitalRead (pinKeyRed))// Перевірка натискання синій чи червоної кнопки.

{ menu = 1 ;} // Повертаємось у подію 1.

digitalWrite (pinBuzzer , HIGH); // Включаємо зумер.

digitalWrite (pinKeyLedBlue , HIGH); // Вмикаємо світлодіод на блакитній кнопці.

delay (100); // Затримка 100 мс.

digitalWrite (pinBuzzer , LOW); // Вимикаємо зумер.

digitalWrite (pinKeyLedBlue , LOW);// Вимикаємо світлодіод на синій кнопці.

```
    delay ( 100 ); // Затримка 100 мс.
 break ; // Виходимо з оператора case.
 }
 }
 void StartGame () // Функція початку гри.
 {
  time . settime ( 0 ); // Задаємо початковий час 0.
  minLeft = startGameTime ; // Привласнюємо задану кількість хвилин лівому гравцю.
  minRight = startGameTime ; // Привласнюємо задану кількість хвилин правому гравцю.
 void ShowTime () // Функція показу часу.
 {
  if ( minLeft >= 0 && minLeft < 10 ) // Якщо кількість часу у лівого гравця від 0 до 9.
 { minLeftChar = ( String ) "0" + minLeft ;} // Додаємо спереду нуль і переводимо в рядкове значення .
 else { minLeftChar = ( String ) minLeft ;} // Інакше просто переводимо кількість часу в рядкове значення.
 if ( minRight >= 0 && minRight < 10 ) // Якщо кількість часу у правого гравця від 0 до 9.
 { minRightChar = ( String ) "0" + minRight ;} // Додаємо спереду нуль і переводимо в рядкове значення.
 else { minRightChar = ( String ) minRight ;} // Інакше просто переводимо кількість часу в рядкове значення. //
  timesMin = minLeftChar + minRightChar ;// Привласнюємо загальною змінною кількість часу лівого та правого гравця.
 if ( micros () - prevmicros > 500000 ) // Кожні пів секунди.
 {
   prevmicros = micros (); // Зберігаємо попереднє значення таймера.
```

counter =! counter ; // Змінюємо змінну події виведення значень на індикатор. if (! counter) // Якщо змінна події false.

```
{
    dispLED . print ( timesMin ); // Виводимо значення індикатор. }
else
{
    dispLED . point ( 2 , true ); // Включаємо другу точку.
}
}
}
```

void ConversionTimeLeft () // Функція додавання часу Фішера для лівого гравця. { //

timeLeft = 59 - timeLeft ; // Визначаємо кількість часу, що залишився лівому гравцю.

timeLeft = timeLeft + fisherTime; // Додаємо час Фішера лівому гравцю.

```
if ( timeLeft > 59 ) // Якщо сумарний час більше 59 у лівого гравця.
{
    timeLeft = timeLeft - 59 ; // Віднімаємо із сумарного часу 59 лівого гравця.
if ( eventLeftZero ) { minLeft = 0 ; eventLeftZero = false ;} // Якщо дозволено виведення секунд на індикатор для лівого гравця, то очищаємо змінну хвилину, забороняємо виведення секунд на індикатор.
    minLeft ++;// Збільшуємо кількість хвилин у лівого гравця.
}
if ( eventLeftZero ){ minLeft = timeLeft ;} // Якщо дозволено виведення секунд на індикатор для лівого гравця, то зберігаємо кількість часу, що залишився лівого гравця в змінну хвилин.
```

timeLeft = 59 - timeLeft ; // Визначаємо зворотну кількість часу, щоб дізнатися час, що залишився для лівого гравця. }

void ConversionTimeRight () // Функція додавання часу Фішера для правого гравця. { //

timeRight = 59- timeRight ; // Визначаємо кількість часу, що залишився правому гравцю.

timeRight = timeRight + fisherTime ; // Додаємо час Фішера правому гравцю.
if (timeRight > 59) // Якщо сумарний час більше 59 у правого гравця.
{

timeRight = timeRight - 59 ; // Віднімаємо із сумарного часу 59 правого гравця.

if (eventRightZero) { minRight = 0 ; eventRightZero =false ;} // Якщо дозволено виведення секунд на індикатор для правого гравця, то очищаємо змінну хвилин, забороняємо виведення секунд на індикатор.

minRight ++; // Збільшуємо кількість хвилин у правого гравця.

}

if (eventRightZero){ minRight = timeRight ;} // Якщо дозволено виведення секунд на індикатор для правого гравця, то зберігаємо кількість часу, що залишився правого гравця в змінну хвилин.

timeRight = 59 - timeRight ; // Визначаємо зворотну кількість часу, щоб дізнатися час, що залишився, для правого гравця.

}

2.9 Опис роботи годинника у зібраному стані

Для початку роботи підключимо живлення Arduino. На індикаторі загориться кількість часу кожного гравця. За замовчуванням: 30 хвилин. Для початку гри потрібно натиснути одну з клавіш: червону або синю. Загориться та кнопка, яка була натиснута. Гравець, що натиснув її, ходить першим. На індикаторі з'являться два числа, розділені точкою по центру. Ліва частина індикатора показує кількість часу гравця, що натискає червону кнопку, права частина індикатора показує кількість часу гравця, що натискає синю кнопку. Після натискання кнопки час, що відноситься до того чи іншого гравця буде зменшуватися, а час суперника буде нерухомо, поки до нього не перейде хід. Якщо зменшуються хвилини, то на індикаторі змінюються

хвилини, якщо зменшуються секунди, то індикатор показуватиме зміни кожну секунду. Після зробленого ходу гравець зобов'язаний ще раз натиснути свою кнопку, щоб його час зупинився, запалилася кнопка суперника, і пішов відлік часу суперника. Після закінчення часу в одного з гравців почне блимати світлодіод на кнопці і пищати зумер, сповіщаючи про програш цього гравця.

2.10 Висновки до розділу 2

У розділі 2 обрано технологію, мову програмування та компоненти АПЗ. Для розроблення програмної частини обрано технологію Arduino та середовище розробки Arduino (IDE), мову програмування Arduino, оскільки у даному проекті використовується плата Arduino Uno. Надано опис використаних функцій. Також обрано необхідні бібліотеки та описано їх структури і функціональність.

У описі інтерфейсів АПЗ представлені схеми підключення компонентів приладу до Arduino. Після чого побудована електрична схема пристрою та розроблена фізична модель приладу для контролю часу. Показано результати збору приладу, загальний вигляд приладу ззовні.

Описан весь функціонал годинника та його можливості. Після тестустування були виявлення деякі помилки у коді та вони були успішно виправлені.

ВИСНОВКИ

Згідно мети бакалаврської роботи розроблено програмно-апаратний комплекс у вигляді годинника Фішера на базі мікроконтролерної плати Arduino.

З аналітичного огляду джерел інформації встановлено, що аналогів на ринку товарів немає для цього годинника. Усі годинники, що продаються на популярних торгових площадок такі як ebay та rozetka мають тільки стандарті годинники для професійних або аматорських турнірів. У порівнянні зі стандартними годинниками, цей проект відрізняється більшою функціональністю та різними видами режимами для гри.

Вибір складових і комплектуючих для розроблення обумовлений співвідношенням їх технічних характеристик та ціни для виготовлення прототипу. Так, для приладу було обрано мікроконтролер Arduino Uno, Trema set shield, Trema-модуль чотирирозрядний LED індикатор, Trema-модуль годинник реального часу, RTC, Trema-модуль кнопку зі світлодіодом 2шт, оскільки це найоптимальніший набір комплектуючих для створення приладу, відповідного бажаним вимогам.

Для розроблення програмної частини обрано технологію Arduino та середовище розробки Arduino (IDE), мову програмування Arduino, оскільки у даному проекті використовується плата Arduino Uno, та компоненти АПЗ.

Побудовано електричну схему приладу та розроблено фізичну модель. Головний результат впровадження розробленого апаратно-програмного комплексу – підвищення якості навчання людей у ситуаціях коли треба швидко приймати рішення, на прикладі такої гри як шахмати. Це сприяє розвитку мислення людей різного віку та являється непоганою заміною для звичайних шахмат. За результатом виконання кваліфікаційною роботи усі поставленні задачі були виконанні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. DS3231 – схема подключения к Arduino. Вольтик. URL : https://inlnk.ru/WD1dv_.

2. Инютин С.А. Метод оценки параметров в системе управления специальной аппаратурой // T-Comm: Телекоммуникации и транспорт. (2015). №1. С. 75-78. URL : https://clck.ru/VQTQa.

3. Среда разработки Arduino. Arduino. URL : https://clck.ru/DXFQ4.

4. Arduino основи програмування. Статті. Geekmatic. URL : http://surl.li/wihf.

5. Thijs Elenbaas. Extension of the standard Arduino EEPROM library. EEPROMEx. Arduino Library List. URL : http://surl.li/wihh.

6. Универсальная библиотека для DHT11 и DHT22, iarduino_DHT. Iarduino. URL : http://surl.li/wihk.

7. Подключение DS3231 к Ардуино по I2C / SPI. Arduino — IoT (интернет вещей). Робототехника Ардуино.URL : https://clck.ru/VQTe2

8. Learn C++ Programminghttps – :// www.programiz.com/cpp-programming

9. Introduction to C++ Programming Language – https://www.geeksforgeeks.org/introduction-to-c-programming-language/

10. What is Arduino UNO? A Getting Started Guide – https://www.rs-online.com/designspark/what-is-arduino-uno-a-getting-started-guide

11. Зуммер із вбудованим генератором - https://iarduino.ru/shop/Expansion-payments/zummer-so-vstroennym-generatorom-trema-modul.html

12. Годинник реального часу, RTC - https://iarduino.ru/shop/Expansion-payments/chasy-realnogo-vremeni-rtc-trema-modul-v2-0.html

13. Кнопка зі світлодіодом синя - https://iarduino.ru/shop/Expansion-payments/knopka-so-svetodiodom-sinyaya-trema-modul.html

14. Кнопка зі світлодіодом червона - https://iarduino.ru/shop/Expansion-payments/knopka-so-svetodiodom-krasnaya-trema-modul.html

15. Чотирирозрядний індикатор LED - https://iarduino.ru/shop/displays/chetyrehrazryadnyy-indikator-led-trema-modul.html

16. Installing dll. Installing and connecting Arduino libraries - https://surgutsto.ru/en/ustanovka-biblioteki-dll-ustanovka-i-podklyuchenie-bibliotek-arduino-ustanovka/

17. iarduino_RTC - https://github.com/tremaru/iarduino_RTC

18. Arduino – https://www.arduino.cc/

19. Бібіотеки Arduino – https://doc.arduino.ua/ru/prog/Libraries

20. Trema slot shield – https://iarduino.ru/shop/Expansion-payments/trema-set-shield.html